TNO FlintFiller: Van Wetstekst naar Flint Frames

Roos Bakker & Maaike de Boer

Versie 2.0; Juli 2020

1. Introductie

Dit document beschrijft de huidige stand van zaken met betrekking tot de FlintFiller, ter informatie voor de projectleden van het project en ter vastlegging voor eventuele vervolgprojecten. De FlintFiller is een tool ontwikkelt door TNO in het kader van het Calculemus-Flint project (2020 -). Het doel van de FlintFiller is om een wetstekst automatisch om te zetten naar een Flint Frame. Het Flint Frame, ontwikkeld door ICTU[1], is een formalisatie van een wetstekst in de vorm van handelingen, feiten en verplichtingen. Een frame over een handeling wordt een *act frame* genoemd. Een voorbeeld van een act frame is te lezen in bijlage 1. Een frame over een feit wordt een *fact frame* genoemd. Een voorbeeld van een fact frame is te lezen in bijlage 2. De frames over verplichtingen worden *duty frames* genoemd. In dit document zullen we niet verder ingaan op de duties, omdat deze nog het minst uitgewerkt zijn.

In de huidige versie van de FlintFiller ligt de focus op wetteksten zoals te vinden op `wetten.nl`[2]. Dit betekent ook dat er momenteel alleen Nederlandse teksten kunnen worden omgezet. In de toekomst zouden ook wetteksten die zich aan een andere standaard, zoals MetaLex[3], houden, of anderstalige teksten kunnen worden omgezet, en uiteindelijk zouden alle teksten die normen bevatten omgezet moeten kunnen worden in Flint frames.

De FlintFiller is ontwikkeld in `Python` (3.7) en de code is te vinden op de Gitlab van Calculemus-Flint[4]. Het TRL (Technology Readiness Level) niveau van deze tool[5] is maximaal 3. Dit betekent dat er een proof-of-concept is die de mogelijkheden voor functionaliteiten weergeeft, maar geenszins inzetbare productieve software is.
De kern van de FlintFiller is gebaseerd op Artificial Intelligence (AI) en specifiek Natural Language Processing. De volgende sectie geeft daarom eerst achtergrondinformatie over Natural Language Processing. Sectie 3 bevat informatie over de huidige versie van de FlintFiller en sectie 4 bevat de conclusie, discussie en suggesties voor de toekomst.

2. Natural Language Processing

Natural Language Processing (NLP) of natuurlijke taalverwerking is een vakgebied dat zich bezighoudt met de vaardigheid van een computerprogramma om menselijke taal te begrijpen. De eerste fundamenten van het vakgebied werden gelegd door Turings (1936) algoritmisch computationeel model, waarna Shannon in 1948 dit voor het eerst toepaste op taal. Chomsky (1956) introduceerde vervolgens de eerste reguliere taal, gegenereerd door een eindige automaat. Dit gaf het startschot voor de ontwikkeling van formele talen en de geïnteresseerde lezer wordt verwezen naar meer detail in (Jurafsky & Martin, 2009). In 1980 werden data gebaseerde algoritmes, zoals machine learning, populairder. De computer kon in theorie aan de hand van voorbeelden automatisch leren. Dit zorgde er onder andere voor dat het automatisch vertalen tussen twee talen (machine translation) mogelijk werd. Dit werd pas populairder vanaf 2000, toen de computers krachtig genoeg werden voor de algoritmes. Vanaf 2010 kwamen *representation learning* en *deep*

[1] https://www.ictu.nl/
[2] https://wetten.overheid.nl/zoeken
[3] http://www.metalex.eu/
[4] https://gitlab.com/calculemus-flint/FlintFiller
[5] https://en.wikipedia.org/wiki/Technology_readiness_level

learning algoritmes op, die voor nieuwe mogelijkheden en betere kwaliteit zorgden op het gebied van NLP. Ondanks dat het vakgebied natuurlijke taalverwerking dus redelijk oud is, zijn nog zeker niet alle taken makkelijk oplosbaar. Daarnaast zijn de ontwikkelingen vooral gericht op de Engelse taal waardoor de ontwikkelingen in en mogelijkheden van de automatische verwerking van het Nederlands daar op achterloopt.

In NLP zijn er verschillende sub-vakgebieden waarvan we er 2 in dieper detail beschrijven. De eerste is *syntax* of syntaxis. De syntax is de structuur van de zinnen. Een voorbeeld van de regels voor het maken van zinnen is grammatica. De vervoegingen van de woorden, zoals werkwoorden, wordt morfologie genoemd, maar wordt in de NLP vaak onder hetzelfde kopje geschaard als syntax. Voorbeelden van taken in de automatische extractie van syntax zijn:

- Sentence Boundary Disambiguation / Sentence Breaking: het omzetten van stukken tekst naar losse zinnen. Punctuatie speelt hierbij een belangrijke rol, maar niet alleen want deze zijn in bijvoorbeeld afkortingen ook aanwezig.
- Word Segmentation: het omzetten van zinnen naar losse woorden. Dit lijkt triviaal door het gebruik van spaties en speciale tekens, maar is het voor sommige talen, zoals het Japans en Chinees, niet.
- Lemmatization: het automatisch verwijderen van de *inflectional ends*; 'lop' voor het werkwoord 'lopen'.
- Stemming: het automatisch omzetten van een woord naar zijn stam; 'loop' voor het werkwoord 'lopen'.
- Part of Speech tagging: het automatisch extraheren van de grammaticale rol van elk woord in de zin; werkwoord, zelfstandig naamwoord, bijvoeglijk naamwoord etc. Vaak worden hiervoor de zgn. *Penn Treebank*[6] tags gebruikt.
- Parsing: het automatisch creëren van een grammaticale boom waarin de relatie tussen de rollen van de woorden ook benoemd is. Zo zijn een lidwoord en zelfstandig naamwoord samen een *noun phrase* (NP) en vormen een werkwoord met een NP een VP (*verb phrase*). Het kan zijn dat zinnen op meerdere manieren kunnen worden gelezen. Deze krijgen dan een andere boom. De syntax bepaalt de mogelijke bomen, maar de 'juiste' boom wordt pas bepaald als de betekenis wordt toegevoegd – de semantiek (zie het volgende subvakgebied). Vaak worden de *universal dependencies*[7] hiervoor gebruikt.

Bij syntax geldt dat voor het Engels de modellen een stuk beter werken dan bij het Nederlands. Over het algemeen gaat het opsplitsen in zinnen en woorden goed, het vinden van de lemma's en stammen redelijk maar zijn de tagging en parsing nog lastig. Voor simpele zinnen die niet ambigue zijn, dus niet op meerdere manieren op te vatten, gaat het over het algemeen goed. Als er lange zinnen of subzinnen zijn met woorden die bijvoorbeeld een vervoegd werkwoord zijn dat ook als zelfstandig naamwoord gebruikt kan worden, wordt de taak al een stuk lastiger. In de FlintFiller zullen in ieder geval sentence breaking, word segmentation, lemmatization of stemming en Part of Speech tagging een rol gaan spelen, welke alle nodig zijn als *pre-processing* (verwerking vooraf) alvorens te kunnen komen tot het het maken van de frames.

Een ander sub-vakgebied in de NLP is *semantics* of semantiek. In de semantiek houdt men zich bezig met de betekenis van een woord of zin. Dit moet niet verward worden met pragmatiek – de betekenis van zin zoals bedoelt door de spreker. Een voorbeeld hiervan is sarcasme; de semantische betekenis is de letterlijke betekenis, terwijl de pragmatische betekenis de figuurlijke is. Ook in de semantiek zijn een aantal taken te onderscheiden, waarvan een aantal voorbeelden:

- Named Entity Recognition (NER): het extraheren van eigennamen, locaties en organisaties. In het Nederlands is dit vaak te herkennen door hoofdlettergebruik.

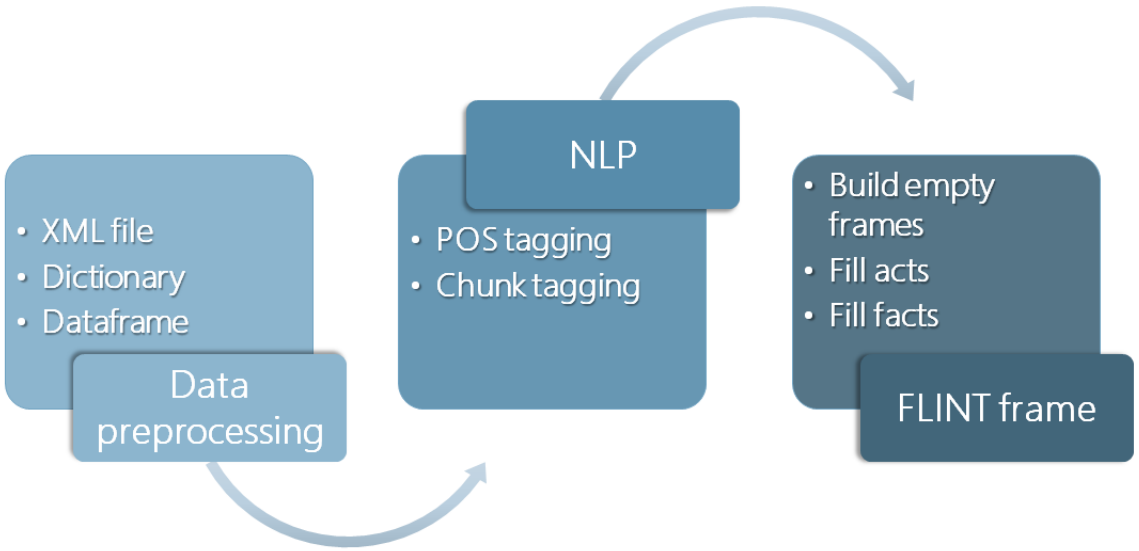
[6] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[7] https://universaldependencies.org/u/dep/

- Sentiment Analysis: het automatisch extraheren van subjectieve informatie
- Topic Modelling: het clusteren van teksten of zinnen die over hetzelfde type onderwerp (topic) gaat
- Machine Translation: het vertalen van tekst uit de ene taal naar een andere taal, met in acht neming van onder andere de grammatica, semantiek en feiten.
- Disambiguation: de meest waarschijnlijke betekenis van een woord of zin in de context van de andere woorden / zinnen kiezen
- Question Answering: gegeven een vraag uit de natuurlijke taal, geef het meest waarschijnlijke antwoord
- Natural Language Generation: het creëren van lopende natuurlijke taal, vaak aan de hand van gestructureerde aanwezige data.
- Natural Language Understanding: het omzetten van natuurlijke taal naar een geformaliseerde taal die door een computer eenvoudig te verwerken is, zoals eerste orde logica.
- Relation(ship) Extraction: het automatisch extraheren van de relaties tussen entiteiten, zoals named entities (x is vader van y).

Semantiek is een stuk lastiger dan syntax. NER, sentiment analyse, topic modelling en machine translations werken redelijk, ook voor het Nederlands. De andere subtaken zijn nog heel lastig om automatisch goed te doen. Zeker de Natural Language Understanding en Relation(ship) Extraction spelen een rol in de FlintFiller, evenals de NER (wie is de actor) en disambiguation.

3. FlintFiller

In Figuur 1 is de architectuur van de FlintFiller te zien. We gaan uit van de wettekst in `XML` zoals te downloaden op `wetten.nl`. Uit deze xml moeten we eerst de elementen halen waarin we geïnteresseerd zijn, zoals de teksten, en dit in een formaat omzetten waar we verdere code op kunnen toepassen. Voor het formaat hebben we gekozen voor een dataframe, een handzame layout van de tekst en meta-informatie in rijen en kolommen wat gelijkenis vertoont met Excel. Het proces om een ruwe wetstekst om te zetten naar een dataframe wordt de *data preprocessing* genoemd in figuur 1. De beschrijving van dit proces is beschreven in sectie 3.1. Een eerste stap het analyseren van de wetsteksten wordt gedaan in het gedeelte van figuur 1 genaamd NLP. Hierin wordt de syntactische – structurele/grammaticale – analyse van de wetteksten gedaan. De beschrijving van dit proces is te lezen in sectie 3.2. In de laatste stap worden de Flint Frames gemaakt. Hierin wordt acts en facts van elkaar onderscheiden, en in de toekomst kunnen hier de duties aan toegevoegd worden. De beschrijving van de creatie van Flint Frames staat in sectie 3.3. De code van deze architectuur is te vinden in de module `FlintFiller` in de gitlab repository.



Figuur 1. FlintFiller Architectuur

3.1 Data preprocessing

Om NLP-algoritmes te kunnen toepassen op teksten, moet de data voorbewerkt worden. Welke stappen hierin genomen moeten worden, verschilt per toepassing, en in het geval van de FlintFiller moet vooral de relevante data uit de xml structuur gehaald worden. Dit klinkt eenvoudiger dan het is: vaak kan je door middel van een xml schema (indien aanwezig) een makkelijke vertaalslag maken naar het formaat wat gewenst is (bijv. een *dictionary* of een dataframe). Helaas is het xml schema van wetten.nl onvolledig en onjuist. Zo is er in het algemene schema van een toestand wel te zien welke elementen deze mogelijk bevat, maar niet hoe de gelaagdheid van de xml eruit ziet. Hierdoor kan helaas geen gebruik worden gemaakt van bestaande XML-parsers en moet deze stap van de preprocessing gerealiseerd worden met op maat gemaakte code.

Eerst transformeren we de xml naar een *dictionary*, de structuur van de wet blijft hierbij hetzelfde. Vervolgens zetten we de dictionary om in een dataframe omdat hiermee data bewerkingen en analyses eenvoudig kunnen worden uitgevoerd, ook met grote hoeveelheden data. De dictionary omzetten in een dataframe is lastig: de structuur van wetten.nl zit complex in elkaar. Zo is er bijvoorbeeld het onderscheid tussen een wetbesluit en een regeling, deze zijn opgezet volgens een verschillende structuur die hieronder beschreven worden.

Voorbewerking van wetbesluiten

Bij het voorbewerken van de data in de FlintFiller zijn we begonnen met ons richten op de Vreemdelingenwet (BWBR0011823, versie 2019-02-27. Deze wet is een wetbesluit, dit is een gedetailleerde beschrijving van de wet waarin staat uitgelegd hoe de wet geïnterpreteerd moet worden. De tekst van deze wat waar we later de NLPanalyse op doen zit verborgen in verschillende lagen, zoals een hoofdstuk, een lid, of een onderdeel. In die lagen zit allerlei informatie die niet over de tekst gaat, zoals verwijzingen, versiedatums, en BWB-nummers. Om de tekst eruit te halen zodat we deze kunnen gebruiken voor NLP-algoritmes, moet alle tekst uit elke laaggeëxtraheerd worden. Belangrijk hierbij is dus om te weten wat voor niveaus er zijn, hoe deze zijn opgebouwd, en op welke plek de tekst zich bevindt, zodat er geen tekst mist.

Een wetbesluit zoals de vreemdelingenwet is opgebouwd uit verschillende lagen: de hoogste laag waar informatie over de wet staat alsmede de aanhef en het wetbesluit zelf. Het wetbesluit zelf

bevat hoofdstukken, deze hoofdstukken bevatten paragrafen en afdelingen. Deze bevatten weer artikelen, en hier bevindt zich soms ook tekst. Vaak staat de tekst ook onder leden en onderdelen van artikelen. De opbouw van deze onderdelen verschilt, en kan een of meerdere onderdelen bevatten die tekst bevatten. Dit houdt in dat onderdelen van verschillende types kunnen zijn (bijvoorbeeld een dictionary of een list), wat het extraheren van alle tekst moeilijker maakt.

Voor het verwerken van de Vreemdelingenwet met de flintfiller hebben we de teksten per artikel samengevoegdIn figuur 2 is een klein deel te zien van het dataframe, met links het bwb-ng-variabel deel nummer, en rechts de tekst van het artikel. Verschillende onderdelen zijn samengevoegd met de speciale tekens '$'.

/Hoofdstuk3/Afdeling3/Paragraaf1/Artikel14	Onze Minister is bevoegd: $$ de aanvraag tot het verlenen van een verblijfsvergunning voor bepaalde tijd in te willigen, af te wijzen dan wel niet in behandeling te nemen; $$ de a... willigen, af te wijzen dan wel niet in behandeling te nemen; $$ een verblijfsv... wijzigen wegens veranderde omstandigheden; $$ een verblijfsvergunning v... wel de geldigheidsduur ervan te verlengen. $$ Onze Minister verleent de ho... verblijf binnen twee weken nadat deze zich overeenkomstig , heeft aangemeld, a... verblijfsvergunning voor bepaalde tijd onder dezelfde beperking als die waaronder de machtiging tot voorlopig verblijf is verleend.artikel 54, eerste lid, o... verblijf binnen twee weken nadat deze zich overeenkomstig , heeft aangemeld, a...

Figuur 2. Dataframe FlintFiller wetbesluit

Voorbewerken van regelingen

Naast de Vreemdelingenwet hebben we ons gericht op het kunnen verwerken van de beleidsregeling Tegemoetkoming Ondernemers Getroffen Sectoren COVID-19 (TOGS)[8]. Dit is een regeling, en deze is anders opgebouwd dan een wetbesluit. Een regeling wordt vaak afgeleid van een wetbesluit en bevat specifieke regels voor een concrete situatie, in dit geval regels omtrent het COVID-19 virus. De structuur van een regeling is anders dan die van een wetbesluit: zo heeft een regeling geen afdelingen en paragrafen, maar wel meer niveaus van onderdelen. Vanwege de gelaagdheid van deze regeling, en de vraag om meer gegevens in het dataframe mee te nemen zoals het opsommingsniveau, versiedatums, etc., hebben we naast het aanpassen van de code op de regeling ook de code herschreven zodat deze beter om kan gaan met verschillende informatie in de wet, en tegelijkertijd generieker is zodat bij nieuwe wetten makkelijk herkend kan worden waarin de structuur mogelijk afwijkt. Een voorbeeld van het nieuwe resulterende dataframe is weergeven in figuur 3.

44	Artikel4	Lid2	Onderdeelh	Onderdeel1				een kopie van een zakelijke huur- of koopovereenkomst van de vestiging; of	22-4-2020
45	Artikel4	Lid2	Onderdeelh	Onderdeel2				een kopie van de belastingaangifte van het jaar 2019 of 2020 waaruit blijkt dat er sprake is van een werkruimte waarvan de vaste lasten en kosten fiscaal aftrekbaar zijn als bedoeld in ;artikel 3.16, eerste lid, van de Wet inkomstenbelasting 2001	22-4-2020
46	Artikel4	Lid2	Onderdeeli					voor zover het een gedupeerde agrarische recreatieonderneming betreft: een verklaring dat het te verwachten omzetverlies, bedoeld in , en de te verwachten vaste lasten, bedoeld in artikel 2, eerste lid, onderdeel b, betrekking hebben op zijn nevenactiviteit met de code 55.20.1, 55.20.2, 55.30 of 93.29.9 van de Standaard Bedrijfsindeling;artikel 2, eerste lid, onderdeel a	22-4-2020
47	Artikel4	Lid2	Onderdeelj					voor zover het een gedupeerde onderneming in de toeleveringsketen betreft: een verklaring dat de onderneming het omzetverlies, bedoeld in , verwacht te lijden doordat de onderneming voor minimaal zeventig procent van zijn omzet afhankelijk is van:artikel 2, eerste lid, onderdeel a	22-4-2020

Figuur 3. Dataframe FlintFiller regeling

In figuur 3 is te zien dat de tekst per onderdeel wordt weergeven, met daarbij ook het onderdeelnummer. Daarnaast is de datum te zien van wanneer de tekst voor het laatst is gewijzigd. Het volledige dataframe en meer technische details over de code zijn te vinden in gitlab.

[8] BWBR0043324, versie 2020-04-22

Lessons Learned

Bij het preprocessen van de wetten.nl teksten zijn we erachter gekomen dat de xml bestanden van wetten.nl niet klopt volgens het xml schema, dat daarnaast niet bij alle wetten een xml schema bekend is, dat de opzet onregelmatig is, en dat er grote verschillen zijn per wet. Dit bij elkaar maakt dat generieke code schrijven om de data te verwerken erg lastig is. Daarom hebben we de code herschreven zodat het relatief makkelijk is om een aanpassing te maken als een wet dat vereist. Dit zorgt ervoor dat gezien het formaat van de wetten xml, de FlintFiller code toch zo generiek is als mogelijk en geschikt is voor hergebruik in de toekomst.

3.2 NLP

In de volgende stap van de FlintFiller pipeline, weergegeven in het midden van figuur 1, richten we ons voornamelijk op syntax - de linguistische structuur van de zinnen die in de wetteksten worden gebruikt. Met het gecreëerde dataframe voorzien we de tekst van annotatie omtrent deze structuur, bijvoorbeeld de zelfstandige naamwoorden en de werkwoorden, want deze informatie is nodig bij het creëren van de Flint Frames.

We gebruiken Part of Speech (POS) tagging – per woord de grammaticale rol aangeven – evenals chunk tagging – een light-weight dependency parsing methode. Met deze twee methoden hebben we een overzicht van de rollen van de woorden per zin, evenals de ruwe relaties tussen de woorden. In de volgende secties beschrijven we de gebruikte methodieken en de lessons learned.

De code is reeds geschreven in het eerste traject van Calculemus-Flint (Jan – Mar 2020). In het tweede traject is er een analyse gedaan van de kwaliteit van de methoden, met name de POS tagging. De code is te vinden in de `chunk_tag_dataframe.py` file.

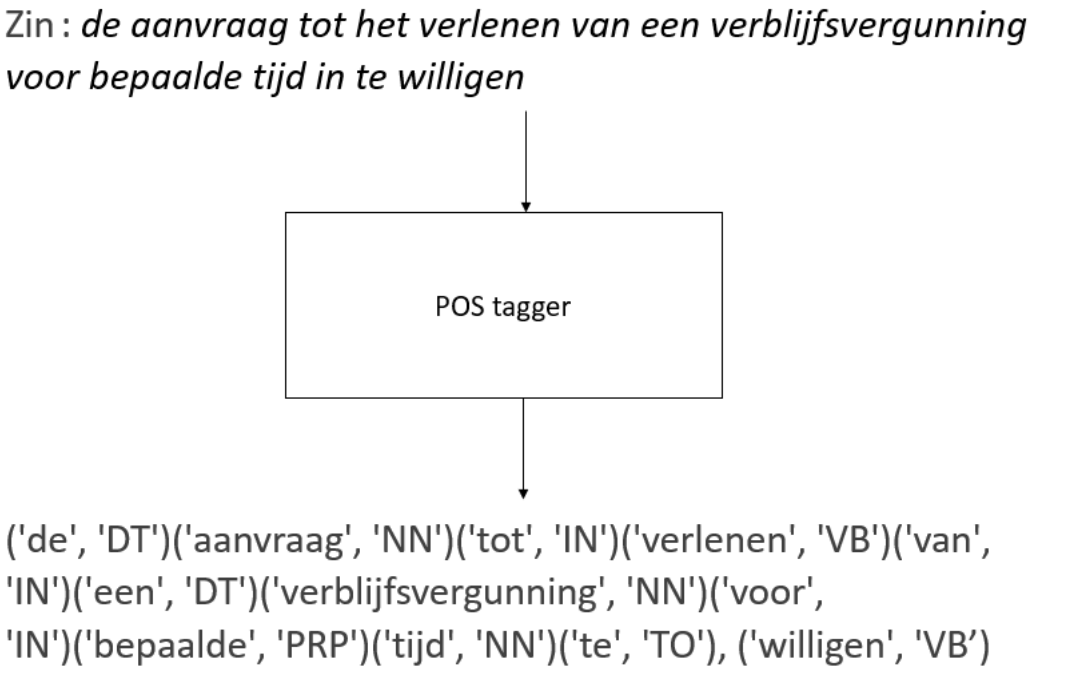
POS tagging

Ondanks dat er legio Part Of Speech tagging implementaties voor het Nederlands te vinden zijn[9], blijken de meesten niet goed te werken op de wetsteksten. Zo blijken veel wet-specifieke woorden onbekend. Als voorbeeld geeft de Stanford Parser bij het werkwoord ‘inwilligen’ None aan als type. Daarnaast worden wetsteksten op een specifieke manier opgebouwd, vaak met bijzinnen waar de POS tagger moeite mee heeft. Dit zorgt er ook voor dat een aantal van de implementaties geen of een foute typering aan de woorden wordt toegekend. Na analyse van de resultaten van een aantal taggers, hebben we voor Pattern[10] gekozen. De resultaten van de POS (en chunk) tagger worden toegevoegd als kolom aan het bestaande Dataframe.

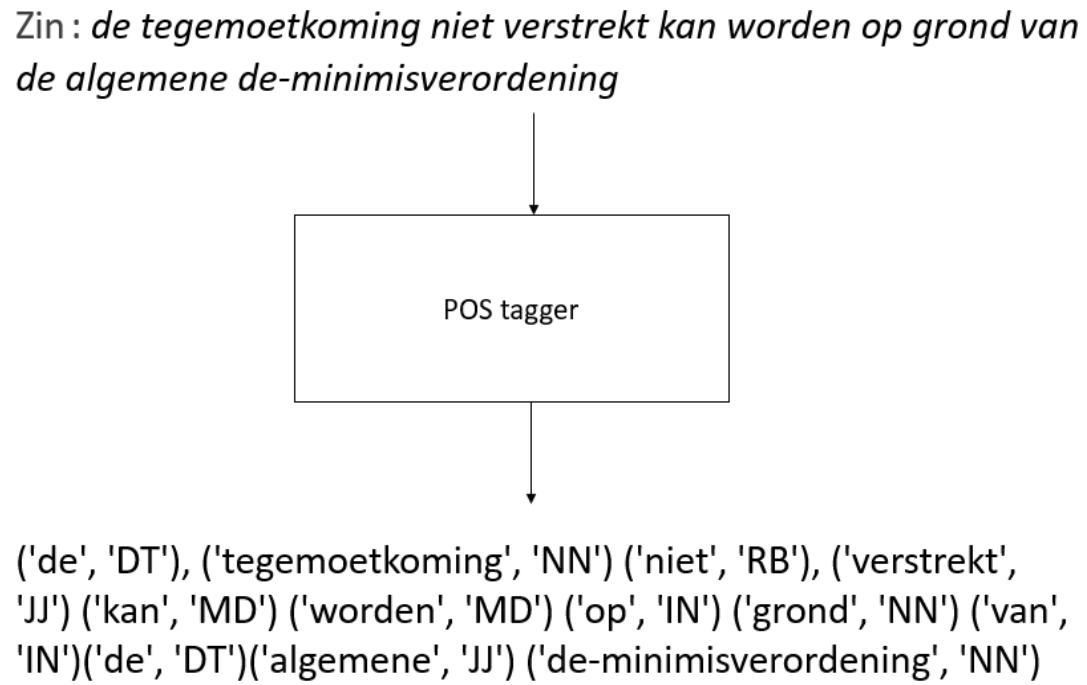
In figuur 4 en 5 zijn twee voorbeelden van bijzinnen te vinden, één uit de Vreemdelingenwet en één uit de TOGS, met de POS tags die daarbij automatisch worden geëxtraheerd. Het valt meteen op dat de zinnen onvolledig zijn en niet op zichzelf kunnen staan.

[9] Frog / LaMachine, SyntaxNet, Spacy, Alpino, Apertium, TextBlob (=Pattern), Dutch-tagger, Pattern, Stanford Parser

[10] https://pypi.org/project/Pattern/



Figuur 4. Resultaat POS tagging zin 1



Figuur 5. Resultaat POS tagging zin 2

Legenda van de genoemde termen:
DT: *determiner*: lidwoord
NN: *noun*: zelfstandig naamwoord
IN: *preposition*: voorzetsel

VB: *verb*: werkwoord
PRP: *personal pronoun*: persoonlijk voornaamwoord
TO: *to*: te
RB: *adverb*: bijwoord
JJ: *adjective*: bijvoeglijk naamwoord
MD: *modal*: modaal (hulp)werkwoord

In de foutenanalyse zoals we die hebben uitgevoerd op de laatste versie van de TOGS bleek dat de tagging over het algemeen goed (genoeg) werkt voor het doeleinde ervan, te weten, het maken van Flint-frames. De volgende structurelere, maar voor het doel niet onoverkomelijke fouten blijven bestaan:

- Soms 'verdwijnen' woorden, zoals 'het' en 'in' in de eerste zin;
- bijvoeglijk naamwoorden worden soms getypeerd als werkwoord (gedupeerde, gestelde) terwijl andere (agrarische, redelijke, algemene) wel juist van de juiste typering worden voorzien;
- uitdrukkingen: 'in overeenstemming met' resulteert erin dat 'met' als werkwoord wordt getypeerd;
- 'beslist' (in: de minister beslist) wordt getypeerd als bijvoeglijk naamwoord
- 'De minister' wordt soms geschreven met hoofdletter, soms met kleine letter. De typering ervan is dan ook verschillend maar wel correct, namelijk een Proper Noun (eigennaam) in het eerste geval en een gewone Noun (zelfstandig naamwoord) in het tweede geval. Dit leidt tot problemen in de volgende stap. Hierbij zou een consistentiealgoritme voorafgaand aan deze stap, een verschil kunnen maken.

Chunk tagging

Chunk tagging is een *light-weight dependency parsing*, waarbij de woorden worden samengenomen in 'chunks', delen of samenstellingen. Zo is er een chunk genaamd NP, een *noun phrase*, welke een combinatie van lidwoord, bijvoeglijk naamwoord en zelfstandig naamwoord kan bevatten. Waar dependency parsing elke relatie in de zin vastlegt, en daardoor computationeel ook vaker meer tijd kost, beschrijft chunk tagging alleen maar drie delen: NP (*noun phrase*), VP (*verb phrase*) en PP (*preposition*). Deze chuncks zijn, net als POS tags, de basiselementen waaruit in de volgende stap de Flint Frames worden gecreeërd. De chunk tags worden geëxtraheerd door middel van dezelfde Pattern methode die voor de POS tags typering gebruikt worden.

Hieronder staan de twee voorbeeldzinnen, met de chunk tags die daarbij automatisch worden geëxtraheerd:

Zin: *de aanvraag tot verlenen van een verblijfsvergunning voor bepaalde tijd in te willigen*

NP	de aanvraag
PP	tot
VP	verlenen
PP	van
NP	een verblijfsvergunning
PP	voor
NP	bepaalde tijd
VP	te willigen

Zin: *de tegemoetkoming niet verstrekt kan worden op grond van de algemene de-minimisverordening*

NP de tegemoetkoming
ADJP niet verstrekt
VP kan
VP worden
PP op
NP grond
PP van
NP de algemene de-minisverordening

In de foutenanalyse gedaan op de laatste versie van de TOGS bleek dat de tagging over het algemeen goed (genoeg) werkt voor de doeleinden. Doordat Pattern voor zowel de POS tags als de chunk tags wordt gebruikt, zullen ook gelijkvormige fouten gemaakt worden: als een werkwoord verkeerd is herkend als POS, zal het ook verkeerd worden geclassificeerd als chunk. In de vergelijk kwam Pattern er voor beide methoden er echter als beste uit, en combinatie met andere methoden zou waarschijnlijk niet heel veel performanceverbetering opleveren.

Lessons Learned

Bij de implementatie van de NLP modules hebben we niet alleen bevestiging gekregen dat Nederlandse taalmodellen slechter werken dan Engelse, maar ook dat wetteksten een 'taal' op zich zijn. Gelukkig kunnen we af met één van de open source modellen en technieken (Pattern), al kostte de evaluatie van de verschillende tools extra tijd.

Ondanks dat er momenteel nog een aantal structurele fouten te vinden zijn, lijken deze geen grote belemming voor het creëren van de Flint Frames. Mocht dit wel het geval zijn, kunnen de structurele fouten eventueel hardcoded verholpen worden.

3.3 Creëren van Flint Frames

Met het volledige dataframe, waarin zowel de POS als chunk tags vanuit de NLP analyse aanwezig zijn, worden de Flint Frames gemaakt. Momenteel is er vooral aandacht besteedt aan de Act Frames en is er een eerste kleine aanzet gemaakt voor de Fact Frames.

Om frames te kunnen creëren moet de stap van syntax – de structuur- van zinnen worden overgegaan naar semantiek – de betekenis van zinnen. Dit is lastig, omdat de computer dan de zin moet begrijpen. Doordat wettelijke zinsconstructen vaak anders zijn dan teksten van andere bronnen, is dit automatisch bijna niet te doen. Toch is er met behulp van regels een eerste aanzet gedaan om de syntactische constructen te gebruiken om de semantische analyse te vergemakkelijken. Idealiter zou er met behulp van Machine Learning / supervised learning een trainingsset aanwezig zijn van wetten met de bijbehorende frames om daar automatisch de regels uit te leren. Helaas is een dergelijke set niet aanwezig, waardoor er gebruik gemaakt moet worden van handmatig gecreëerde regels. Door via de syntax te werken, worden de regels algemener en kunnen we in de toekomst ook omgaan met nieuwe woorden die nog niet eerder in een wet zijn voorgekomen. Zo zijn de regels gebaseerd op de specifieke (grammaticale) rol van woorden in een zin, en niet op een specifiek woord, wat ook zou resulteren in veel meer regels.

Net als voor NLP geldt dat hier vooral in de eerste sprint aan is gewerkt, en dat de analyse van de resultaten in de tweede sprint is gedaan. Doordat er in de eerste sprint ook hard is gewerkt aan het creëren van het dataframe en de NLP, heeft in het creëren van de Flint Frames relatief weinig tijd gezeten. Hierdoor zijn er nog veel stappen voor verbetering mogelijk, welke ook in de suggesties voor de toekomst zijn opgenomen.

Act Frames

Het creëren van de act frames begint met het detecteren van zinnen die mogelijk een act bevatten. Aan de hand van een lijst van actiewerkwoorden - geverifieerd met een expert - worden deze zinnen eruit gefilterd. Doordat de werkwoorden in allerlei vervoegingen aanwezig kunnen zijn, wordt elk woord dat als een werkwoord is geclassificeerd door de POS tagger, vervoegd naar de infinitieve vorm. Ook dit wordt gedaan middels het Pattern package in Python.

Hieronder staan twee voorbeeldzinnen, met de geëxtraheerde infinitieven:

Zin: *de aanvraag tot het verlenen van een verblijfsvergunning voor bepaalde tijd in te willigen*

Verlenen -> verlenen
In te willigen -> willigen

Doordat inwilligen onbekend is in het vocabulaire van het Pattern package, wordt deze helaas niet goed herkend.

Zin: *de tegemoetkoming niet verstrekt kan worden op grond van de algemene de-minimisverordening*

kan -> kunnen
worden -> worden

De omzetting van verstrekt naar verstrekken gaat helaas niet goed, omdat verstrekt was geclassificeerd als een adjective in plaats van een werkwoord. De gevonden infinitieven worden toegevoegd als extra kolom in het dataframe. De code voor de infinitieven is te vinden in *chunk_tag_dataframe.py*.

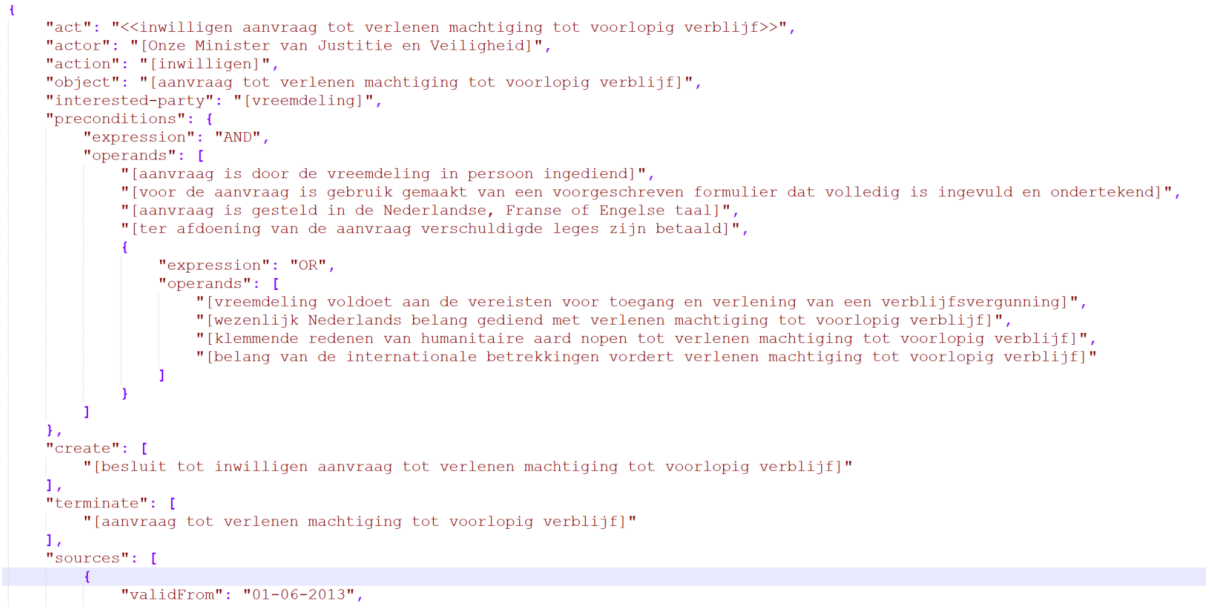
Met de toegevoegde infinitieven wordt er per zin gekeken of er een actiewerkwoord in voorkomt. Als dit het geval is, wordt er een leeg act frame aangemaakt. Vervolgens wordt de act frame aan de hand van de volgende regels gevuld:

- Action: actiewerkwoord (infinitieve vorm)
    - Uitzondering: indien -> indienen is niet de bedoeling; deze wordt eruit gefilterd. Ook herkende werkwoorden waar een lidwoord (de, het of een) voorstaat, wordt gefilterd.
- Precondition: als er een opsomming volgt, haal deze uit elkaar en maak er operands van. De expression is standaard AND.
- Object: het eerste NP in de zin waarin het VP het actiewerkwoord bevat
- Actor: een woord met een tag die relateert aan een Proper Noun (PRP($), NNPS(S))
    - Hierbij zijn ook uitzonderingen gemaakt: Onderdeel, Lid, Indien, Tenzij, Onverminderd, Nadat
- Act: combinatie van action + object
- Recipient (hiervoor: interested-party): ‘vreemdeling’ als dat in de zin voorkomt; nog geen verdere regel. Vaak is dit niet het lijdend voorwerp in de zin.
- Create: geen implementatie
- Terminate: geen implementatie
- Sources:
    - Valid_from: versie
    - Citation: art. (artikel + lid), naam_van_de_wet
    - Text: originele stuk tekst
    - Juriconnect: jci1.3

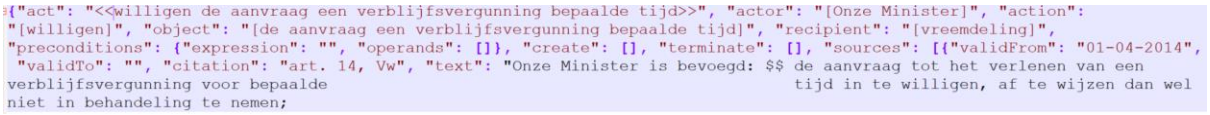
De code is te vinden in *dataframe_to_frame_parser.py*.

Hieronder zijn een aantal screenshots te vinden van de gekozen zinnen en de gecreëerde act frames (figuur 6 - 9). Het bovenste figuur is de handmatige gecreëerde frame door een expert, en het onderste figuur is de automatisch gecreëerde frame. Het label van de frames kan net anders zijn, omdat deze nog in ontwikkeling waren.

Zin: *de aanvraag tot het verlenen van een verblijfsvergunning voor bepaalde tijd in te willigen*

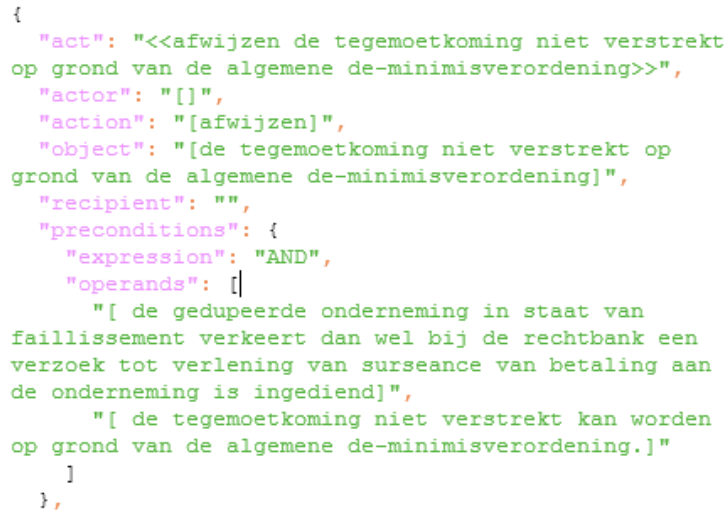


Figuur 6. Handmatige gecreëerd act frame voor zin 1



Figuur 7. Automatisch gecreëerd act frame voor zin 1

Zin: *de tegemoetkoming niet verstrekt kan worden op grond van de algemene de-minimisverordening*



Figuur 8. Handmatige gecreëerd act frame voor zin 2


```
{
  "act": "<<afwijzen de tegemoetkoming niet verstrekt op grond van de algemene de-minimisverordening>>",
  "actor": "[]",
  "action": "[afwijzen]",
  "object": "[de tegemoetkoming niet verstrekt op grond van de algemene de-minimisverordening]",
  "recipient": "",
  "preconditions": {
    "expression": "AND",
    "operands": [
      "[ de gedupeerde onderneming in staat van faillissement verkeert dan wel bij de rechtbank een verzoek tot verlenin
      "[ de tegemoetkoming niet verstrekt kan worden op grond van de algemene de-minimisverordening.]"
    ]
  },
  "create": [],
  "terminate": [],
  "sources": [
    {
      "validFrom": "31-03-2020",
      "citation": "art. 3, TOGS",
      "text": "De minister beslist afwijzend op een aanvraag indien:  $ Onderdeela :  $  de aanvraag niet voldoet aan de
      "juriconnect": "jci1.3:c:BWBR0043324&artikel=3&z=2020-03-31&g=2020-03-31"
    }
  ],
  "explanation": ""
},
```


Figuur 9. Automatisch gecreëerd act frame voor zin 2

In de foutenanalyse gedaan op de laatste versie van de TOGS bleek dat het herkennen van de actiewerkwoorden in principe altijd goed gaat, vanwege de handgemaakte lijst met acties. De fouten die voorkomen zijn fouten als resultaat van de verkeerde POS tagging, zoals:

- In -> innen
- Betreffende / betreft -> betreffen

In beide gevallen is het niet gewenst om een act frame te baseren op deze termen.

In de act frames is er een analyse per onderdeel gedaan. Een aantal onderdelen gaat goed, een aantal is lastig:

- Het werkwoord is een basis om een act frame te maken. Hierdoor gaat de action altijd goed.
- Object: gaat vaak goed, maar kan soms een hele lange zin zijn.
- Actor: is lastig. Als er hoofdletters worden gebruikt, gaat het goed, maar soms worden onlogische actoren gevonden. Ook is de actor niet altijd direct in die zin aanwezig.
- Recipient: is lastig. Nu alleen mogelijk als het in de zin genoemd wordt, of anders moet het uit de fact naar voren komt. De relaties met andere zinnen maken is nog niet aanwezig.
- Act: gaat goed als de action en het object kloppen
- Preconditions / create / terminate: zijn heel lastig automatisch, gaat vaker niet goed dan wel
- Citation / valid from / text: komen vanzelf uit de dataframe; gaat meestal goed.

Fact Frames

Fact Frames gaan over feiten die waar moeten zijn, en zijn een basis voor bijvoorbeeld preconditions. Momenteel is er alleen gekeken naar zinnen die expliciet een definitie bevatten, en nog niet naar ander type zinnen die ook een feit kunnen zijn. Hierdoor zal een gecreëerd flint frame in de Flint Editor van ICTU mogelijk ook foutmeldingen geven, omdat er feiten worden genoemd in acts die niet in de facts aanwezig zijn.

De geëxtraheerde feiten bestaan momenteel alleen uit feiten genoemd in artikel 1 van een wet, waarbij er ook een : in de zin staat. Als dit is gedetecteerd worden de volgende regels toegepast:

- Fact: gedeelte voor de :
- Function: gedeelte na de : (met automatisch een AND als operands); Dit is dezelfde code als voor de acts

- Sources: dit is dezelfde code als voor de acts

De code is te vinden in *dataframe_to_frame_parser.py*.

Hieronder zijn een aantal screenshots te vinden van de definitie van een vreemdeling en de gecreëerde act frames. Het bovenste figuur is de handmatige gecreëerde frame door een expert, en het onderste figuur is de automatisch gecreëerde frame. Het label van de frames kan net anders zijn, omdat deze nog in ontwikkeling waren.


```
{
    "fact": "[vreemdeling]",
    "function": "[]",
    "sources": [
        {
            "validFrom": "01-01-2017",
            "validTo": "31-12-9999",
            "citation": "art. 1 Vw",
            "juriconnect": "jci1.3:c:BWBR0011823&hoofdstuk=1&afdeling=1&artikel=1&z=2017-01-01&g=2017-01-01",
            "text": "vreemdeling: ieder die de Nederlandse nationaliteit niet bezit en niet op grond van een wettelijke bepaling als Nederlander moet worden
            behandeld",
            "explanation": "NOTABLE: De structuur van dit artikel is meerdere keren veranderd sinds april 2001. Ook de plaats van dit fragment in het artikel
            verandert nogal eens. Maar de tekst zelf is vanaf de eerste versie niet veranderd. De juriconnect referentie verwijst naar de versie van 2002-01-01
            omdat dat de eerste versie is waarvoor een link bestaat."
        }
    ],
    "explanation": ""
},
```


Figuur 10. Handmatige gecreëerd fact frame


```
{"fact": "[vreemdeling]", "function": {"expression": "AND", "operands": ["[ieder die de Nederlandse nationaliteit niet bezit en niet op
grond                                   van een wettelijke bepaling als Nederlander moet worden behandeld;]"]}, "sources": [{"validFrom":
"28-07-2018", "validTo": "", "citation": "art. 1, Vw", "text": " vreemdeling: ieder die de Nederlandse nationaliteit niet bezit en niet op
grond                                   van een wettelijke bepaling als Nederlander moet worden behandeld;", "juriconnect":
"jci1.3:c:BWBR0011823&hoofdstuk=1&afdeling=1&artikel=1&z=2019-02-27&g=2019-02-27"}], "explanation": ""}]
```


Figuur 11. Automatisch gecreëerd fact frameIn de foutenanalyse gedaan op de laatste versie van de TOGS bleek dat het creëren van fact frames in de basis goed gaat, maar dat het zeker nog niet compleet is. Zinnen die niet expliciet een feit bevatten, zijn lastig te herkennen. Hier zal een kip-ei probleem ontstaan, omdat om de pre-conditions te kunnen herkennen het zou helpen omgoede feiten te hebben om uit te selecteren, terwijl als de pre-conditions zelf bestaan, de feiten makkelijker volgen.

Lessons Learned

In het creëren van de Flint Frames hebben we geleerd dat:

- Een goede analyse van de frames en de denkstappen naar het creëren van frames nodig is om de Flint Frames te begrijpen (als mens)
- Sommige stukken Flint Frames over zinnen of zelfs onderdelen heen gaan; een actor of recipient kan in een onderdeel niet voorkomen, maar toch impliciet die rol hebben
- Het automatisch creëren van Flint Frames op basis van alleen NLP een hele uitdaging zal worden
    - Veel informatie is impliciet
    - De taalconstructen zijn niet altijd consistent / er zijn veel uitzonderingen
    - Taalconstructen in wetten zijn heel specifiek

4. Conclusie, Discussie en Suggesties voor de Toekomst

In dit document hebben we de huidige stand van zaken met betrekking tot de TNO FlintFiller beschreven. Momenteel is het TRL niveau van de FlintFiller nog maximaal 3, waardoor de FlintFiller zeker nog niet af of beschikbaar is om al in te zetten

Hieronder staat kort beschreven wat de huidige versie van de FlintFiller wel kan, en wat nog niet.

Wat kan de FlintFiller nu wel:

- Wetten automatisch verwerken en analyseren (zie sectie 3.1 en 3.2)

        - Dit werkt op de vreemdelingenwet,
        - **Maar ook op andere wetten!** (getest op o.a. grondwet, algemene wet bestuursrecht, comptabiliteitswet en aanwijzingen voor de regelgeving)
        - Het werkt op wetten die volgens het regelingformat en wetbesluitsformat zijn opgezet. Dit dankzij specifiek hiervoor gecreeërde tekst parsing code.
        -
- Wetteksten ontleden
    - Dit zijn de POS en Chunk tags en de infinitieven
    - Automatisch invullen van het FLINT frame
    - Acts
    - Facts

Wat kan de FlintFiller nu niet:

1) Wetten parsen en analyseren
    a) Wetten van andere bronnen analyseren. Dit ligt aan het specifieke format waarmee wetten.nl werkt. Om dit generiek toepasbaarder te maken is het noodzakelijk dat de wetteksten valideren tegen het XML schema (oftewel, dat het XML-schema volledig en correct is)

2) Wetteksten ontleden
    a) Niet alle werkwoorden worden op de juiste manier herkend ('het verlenen' is geen werkwoord)

3) Automatisch invullen van het FLINT frame
    a) Eerste stappen zijn gezet, maar nog niet alle velden gaan goed; preconditions zijn moeilijk, omdat deze vaak over zinnen of artikelen heen gaan.

Een aantal suggesties voor verbetering in de toekomst zijn:

1. Human in the Loop: 2 opties:
    a. Stap voor stap gebruiker meenemen in het proces van het creëren van frames; eerste de POS tags controleren, dan de chunks, vervolgens de acts etc.
    b. Alleen tonen wat zeker is en de rest door de gebruiker laten invullen / annoteren (eventueel met opties voor actors als we die hebben) (en dan ook zinnen tonen waar niets in zit, zodat de gebruiker eventueel een frame kan toevoegen)
2. Om kunnen gaan met een nieuwe versie van de wet
    a. Zo automatisch mogelijk aangeven welke frames veranderd moeten worden
    b. Mogelijk al een eerste suggestie voor verandering doen
3. Huidige FlintFiller verbeteren
    a. Toevoegen van versie van de wet
    b. Facts detecteren
    c. Verbeteren van de actiewerkwoorden om de act frames beter op te kunnen zetten
    d. Het automatisch ontdekken van de Actor + Recipient verbeteren
    e. Kijken naar een vervolgstap in het uitwerken van preconditions
4. Het schrijven van een user story vanuit het oogpunt van een jurist die Flint-frames maakt, ondersteund door de FlintFiller (en eventueel andere tools uit de CF-toolbox).
5. Het iteratief ontwikkelen van een user interface om die user story te realiseren.
6. Het waar nodig aanpassen van de Flintfiller (en andere tools) n.a.v. de inzichten uit de iteraties.

7. Het formaliseren en inzetten van kennis om de FlintFiller te ondersteunen bij lastigere tekstfragmenten, en de detectie van impliciete informatie in de wettekst (dus syntactisch afwezig maar semantisch aanwezig) mogelijk te maken
8. Het ondersteunen van de ontwikkelaars van de oorspronkelijke wet-xml en het xml schema, deze sluiten nu niet op elkaar aan en dit zou een waardevolle ontwikkeling zijn als dat wel het geval is.

Verwijzingen

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory, 2*(3), 113-124.

Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing.* New Jersey: Pearson Education International.

Shannen, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal, 27*(3), 379-423.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *J. of Math, 58*(5), 345-363.

Bijlage 1: Act Frame


```
{
    "act": "<<inwilligen aanvraag tot verlenen machtiging tot voorlopig verblijf>>",
    "actor": "[Onze Minister van Justitie en Veiligheid]",
    "action": "[inwilligen]",
    "object": "[aanvraag tot verlenen machtiging tot voorlopig verblijf]",
    "interested-party": "[vreemdeling]",
    "preconditions": {
      "expression": "AND",
      "operands": [
        "[aanvraag is door de vreemdeling in persoon ingediend]",
        "[voor de aanvraag is gebruik gemaakt van een voorgeschreven formulier dat volledig is ingevuld en ondertekend]",
        "[aanvraag is gesteld in de Nederlandse, Franse of Engelse taal]",
        "[ter afdoening van de aanvraag verschuldigde leges zijn betaald]",
        {
```

```
            "expression": "OR",
            "operands": [
              "[vreemdeling voldoet aan de vereisten voor toegang en verlening van een verblijfsvergunning]",
              "[wezenlijk Nederlands belang gediend met verlenen machtiging tot voorlopig verblijf]",
              "[klemmende redenen van humanitaire aard nopen tot verlenen machtiging tot voorlopig verblijf]",
              "[belang van de internationale betrekkingen vordert verlenen machtiging tot voorlopig verblijf]"
            ]
          }
        ]
      },
      "create": [
        "[besluit tot inwilligen aanvraag tot verlenen machtiging tot voorlopig verblijf]"
      ],
      "terminate": [
        "[aanvraag tot verlenen machtiging tot voorlopig verblijf]"
      ],
      "version": "1-[20130601]-[jjjjmmdd]",
      "reference": "art. 2k, aanhef en onder a, Vw",
      "juriconnect": "jci1.3:c:BWBR0011823&hoofdstuk=1a&afdeling=1&paragraaf=3&artikel=2k&z=2013-06-01&g=2013-06-01",
      "sourcetext": "Onze Minister is bevoegd: de aanvraag tot het verlenen van een machtiging tot voorlopig verblijf dan wel terugkeervisum in te willigen, af te wijzen dan wel niet in behandeling te nemen",
      "explanation": "NOTABLE: Deze clause is op 11-08-2008 gepubliceerd als art. 2c Vw en trad op 01-06-2013 in werking als art. 2k."
    }
```


Bijlage 2: Fact Frame


```
{
    "fact": "[vreemdeling voldoet aan voorwaarden voor het verlenen van een vvr-bep]",
```

```
"function": {
  "expression": "AND",
  "operands": [
    "[doel waarvoor het verblijf is toegestaan]",
    "[vreemdeling beschikt over een geldige machtiging tot voorlopig verblijf]",
    "[vreemdeling beschikt over een geldig document voor grensoverschrijding]",
    "[vreemdeling beschikt zelfstandig en duurzaam over voldoende middelen van bestaan]",
    {
      "expression": "NOT",
      "operand": "[vreemdeling vormt een gevaar voor de openbare orde of nationale veiligheid]"
    },
    "[vreemdeling is bereid om medewerking te verlenen aan een medisch onderzoek naar een ziekte aangewezen bij of krachtens de Wpg of een medische behandeling tegen een dergelijke ziekte te ondergaan]",
    "[vreemdeling voor een werkgever arbeid heeft verricht, zonder dat aan de Wet arbeid vreemdelingen is voldaan]",
    "[vreemdeling voldoet aan de beperking, verband houdende met het doel waarvoor hij wil verblijven]",
    "[vreemdeling beschikt over kennis op basisniveau van de Nederlandse taal en de Nederlandse maatschappij]",
    {
      "expression": "NOT",
      "operand": "[vreemdeling heeft onjuiste gegevens verstrekt dan wel gegevens achtergehouden]"
    },
    {
      "expression": "NOT",
      "operand": "[vreemdeling heeft in Nederland verblijf gehouden, anders dan op grond van art. 8 Vw]"
    },
    "[ten behoeve van het verblijf van de vreemdeling is een verklaring van een referent overgelegd als bedoeld in art. 2a lid 1 Vw]",
    {
```

```
                "expression": "NOT",
                "operand": "[vreemdeling is leges verschuldigd terzake van de afdoening van een aanvraag]"
            }
          ]
        },
        "version": "1-[20010401]-[jjjjmmdd]",
        "reference": "art. 26 lid 1 Vw",
        "juriconnect": "jci1.3:c:BWBR0011823&hoofdstuk=3&afdeling=3&paragraaf=5&artikel=26&z=2002-01-01&g=2002-01-01",
        "sourcetext": "",
        "explanation": "QUESTION: Hoe omgaan met het gebruik van 'de' en 'een' in FLINT. Bijvoorbeeld is [vreemdeling is leges verschuldigd terzake van de afdoening van een aanvraag]: 'de' of 'een' aanvraag?"
    }
```