

Scholieren met dyslexie van de derde graad middelbaar onderwijs ondersteunen bij het intensief lezen van wetenschappelijke artikelen via geautomatiseerde en gepersonaliseerde tekstvereenvoudiging.

Dylan Cluyse.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Mevr. L. De Mol

Co-promotor: J. Decorte; J. Van Damme;

Academiejaar: 2022–2023

Eerste examenperiode

Departement IT en Digitale Innovatie .



Woord vooraf

Deze scriptie en het bijhorende onderzoek zou niet tot stand zijn gekomen zonder de waardevolle bijdragen van diverse individuen die mij hebben ondersteund en gestimuleerd tijdens mijn onderzoek. Ik wil graag mijn oprechte dank betuigen aan deze personen.

Ten eerste, wil ik mijn promotor Lena De Mol bedanken voor haar uitmuntende begeleiding tijdens het onderzoek. Haar affiniteit voor technologie, taal en onderwijs vormde een perfecte match met het onderzoeksgebied van deze scriptie. Daarnaast wil ik graag Johan Decorte en Jana Van Damme bedanken voor hun deskundige inbreng op de vakgebieden machinaal leren en logopedie. Elke wekelijkse sessie met Johan bracht nieuwe inzichten in hoe ik het technologische component van mijn onderzoek kon aanpakken. Dit heeft mijn ambitie alleen maar vergroot. Jana ben ik dankbaar voor haar begeleiding en follow-up op het gebied van logopedie. Haar expertise heeft mijn horizon verbreed binnen dit vakgebied. Verder wil ik Emmanuel Vercruyse en Johannes Nijs van Hogeschool Vives en Sofie Smet en Sophie Vyncke van Arteveldehogeschool bedanken voor hun bijdragen aan de referentieteksten voor het experiment. Alle lectoren hebben mij met veel plezier geholpen door deze taken op zich te nemen, ondanks hun drukke agenda's. Tot slot, wil ik mijn goede vriendin Lobke bedanken voor haar constante steun en aanmoediging tijdens het hele onderzoeksproces, alsook mijn grootste steunpunt die ik tijdens het schrijven van deze scriptie heb kunnen vinden.

Ik wil graag benadrukken dat deze personen van onschatbare waarde zijn geweest voor het succes van mijn onderzoek en het eindresultaat. Hun inzet en toewijding hebben ertoe bijgedragen dat ik deze scriptie met trots kan presenteren.

Samenvatting

Ingewikkelde woordenschat en zinsbouw hinderen scholieren met dyslexie in het derde graad middelbaar onderwijs bij het lezen van wetenschappelijke artikelen. Gepersonaliseerde tekstvereenvoudiging helpt deze scholieren bij hun leesbegrip. Daarnaast kan artificiële intelligentie (AI) dit proces automatiseren om de werkdruk bij leraren en scholieren te verminderen. Dit onderzoek achterhaalt met welke technologische en logopedische aspecten AI-ontwikkelaars rekening moeten houden bij de ontwikkeling van een AI-toepassing voor geautomatiseerde en gepersonaliseerde tekstvereenvoudiging. Hiervoor is de volgende onderzoeksraag opgesteld: "Hoe kan een wetenschappelijk artikel automatisch worden vereenvoudigd, gericht op de unieke noden van scholieren met dyslexie in het derde graad middelbaar onderwijs?". Een requirementsanalyse achterhaalt de benodigde functionaliteiten om gepersonaliseerde en geautomatiseerde tekstvereenvoudiging mogelijk te maken. Vervolgens wijst de vergelijkende studie uit welk taalmodel kan worden ingezet om de taak van gepersonaliseerde en geautomatiseerde tekstvereenvoudiging mogelijk te maken. De requirementsanalyse wijst uit dat toepassingen om wetenschappelijke artikelen te vereenvoudigen, gemaakt zijn voor een centrale doelgroep en geen rekening houden met de unieke noden van een scholier met dyslexie in het derde graad middelbaar onderwijs. Adaptieve software voor geautomatiseerde tekstvereenvoudiging is mogelijk, maar ontwikkelaars moeten meer inzetten op de unieke noden van deze scholieren.

Inhoudsopgave

Lijst van figuren	vii
Lijst van tabellen	ix
1 Inleiding	1
1.1 Probleemstelling	2
1.2 Onderzoeksvraag	3
1.3 Onderzoeksdoelstelling	5
1.4 Opzet van deze bachelorproef	5
2 Stand van zaken	6
2.1 Inleiding	6
2.2 Specifieke noden en richtpunten	6
2.2.1 Specifieke noden van scholieren met dyslexie in de derde graad van het middelbaar onderwijs.....	7
2.2.2 Specifieke kenmerken van wetenschappelijke artikelen	8
2.3 Aanpakken voor tekstvereenvoudiging.....	12
2.3.1 Manuele tekstvereenvoudiging	12
2.3.2 Bevoordelende effecten van MTS bij scholieren met dyslexie ..	13
2.3.3 Aanpak voor ATS.....	16
2.4 De verschillende soorten ATS	18
2.5 Beschikbare tools en taalmodellen	24
2.6 De valkuilen bij AI en NLP.	35
2.7 Conclusie	38
3 Methodologie	39
3.1 Requirementsanalyse	39
3.2 Vergelijkende studie	45
3.3 Prototype voor tekstvereenvoudiging.....	54
3.3.1 De ontwikkeling van het scholierencOMPONENT.....	70
3.3.2 De opzet voor een lokale webtoepassing.....	73
3.3.3 Experimenten en vergelijkende studie met het prototype.	74
4 Resultaten	75
4.1 Ontbrekende functies bij AI-toepassingen	75
4.2 Geschikte taalmodel voor gepersonaliseerde tekstvereenvoudiging met ATS.....	77

4.3 Het prototype voor tekstvereenvoudiging met ATS vergeleken met top-of-the-line tools.	86
5 Conclusie	93
6 Discussie	95
A Onderzoeksvoorstel	98
A.1 Introductie	98
A.2 State-of-the-art	100
A.2.1 Tekstvereenvoudiging	100
A.2.2 Noden van scholieren met dyslexie	100
A.2.3 Huidige toepassingen	102
A.2.4 Ontwikkelen met AI	102
A.3 Methodologie	103
A.4 Verwacht resultaat, conclusie	105
B Referentieteksten: Richtlijnen	106
B.0.1 Lexicale vereenvoudiging	108
B.0.2 Syntactische vereenvoudiging	108
B.0.3 Samenvatten	108
B.1 Specifieke richtlijnen voor A1	109
B.2 Specifieke richtlijnen voor A2	110
C Figures: Requirementsanalyse	112
D Code: Ontwikkeling Van Het Prototype	115
Bibliografie	127

Lijst van figuren

1.1 Het leesplezier bij 15-jarigen volgens de PISA-test (De Meyer e.a., 2019).	2
2.1 De toename van vereiste leesgraad volgens FRE (links) en NDC (rechts). Bron: (Plavén-Sigray e.a., 2017)	11
2.2 De gemeten <i>mean fixation duration</i> tijdens begrijpend lezen uit het onderzoek van Rello, Baeza-Yates, Dempere-Marco e.a. (2013).	14
2.3 Voorbeeld van PoS-labeling uit het artikel van Bilici (2021).	18
2.4 Een pipeline voor LS volgens Althunayyan en Azmi (2021).	19
2.5 Afbeelding van Gooding 2022	28
2.6 Afbeelding van Gooding 2022	29
2.7 Afbeelding van Gooding 2022.	29
2.8 Een experiment op de <i>mean-regret</i> van GPT-3 engines uit Binz en Schulz (2023).	32
2.9 De werking van <i>prompt engineering</i> volgens McFarland (2023)	33
2.10 De evolutie van LLM's (Simon, 2021)	35
3.1 Het benodigde stappenplan bij de requirementanalyse.	40
3.2 Het gevolgde stappenplan voor de vergelijkende studie.	46
3.3 Algemeen overzicht van de ontwikkeling van het prototype voor ATS van wetenschappelijke artikelen.	56
4.1 Overzicht van het minimum, maximum en gemiddeld aantal woorden per zin per model in A1.	79
4.2 Overzicht van het minimum, maximum en gemiddeld aantal woorden per zin per model in A2.	79
4.3 Boxplot van de FRE-scores voor A1.	80
4.4 Boxplot van de FRE-scores voor A2.	80
4.5 Boxplot van de FOG-scores voor A1.	81
4.6 Boxplot van de FOG-scores voor A2.	81
4.7 Gemiddeld aantal complexe woorden per zin gegroepeerd op model voor A1.	82
4.8 Gemiddeld aantal complexe woorden per zin gegroepeerd op model voor A2.	82
4.9 Gemiddeld aantal lange woorden per zin gegroepeerd op model voor A1.	83

4.10 Gemiddeld aantal lange woorden per zin gegroepeerd op model voor A2.	83
4.11 Gemiddeld aantal hulpwerkwoorden per zin gegroepeerd op model voor A1.	84
4.12 Gemiddeld aantal hulpwerkwoorden per zin gegroepeerd op model voor A2.	84
4.13 Gemiddeld aantal vervoegingen van het werkwoord 'zijn' per zin gegroepeerd op model voor A1.	85
4.14 Gemiddeld aantal vervoegingen van het werkwoord 'zijn' per zin gegroepeerd op model voor A2.	85
4.15 Een mogelijke weergaven van de homepagina.	87
4.17 Een mogelijke weergave van het lerarencomponent met het wetenschappelijk artikel van Van Brakel (2022) als input.	88
4.16 Voorbeeldweergave van de instellingenpagina.	88
4.18 Een voorbeeldweergave van het scholierencomponent.	89
4.19 Een voorbeeldweergave van de toepassing van PoS-tagging bij het scholierencomponent.	89
4.20 Stap 1 van een gepersonaliseerde tekstvereenvoudiging in het scholierencomponent.	90
4.21 Stap 3 van een gepersonaliseerde tekstvereenvoudiging in het scholierencomponent.	90
4.22 Stap 1 bij het stellen van een specifieke vraag bij gemarkeerde tekst. .	91
4.23 Stap 2 bij het stellen van een specifieke vraag bij gemarkeerde tekst. .	91
A.1 (Readable, 2021)	104
C.1 Informatie opvragen van een wetenschappelijk artikel met SciSpace .	112
C.2 Tekstvereenvoudiging via de link van een wetenschappelijk artikel met Bing Chat	113
C.3 Illustratie van de tekstanalyse bij Simplish na een tekstvereenvoudiging.	113
C.4 Illustratie van de tekstanalyse bij Rewordify.	114

Lijst van tabellen

2.1	Specifieke drempels bij het begrijpend lezen van een tekst.	7
2.2	Oplossingen die software-ontwikkelaars kunnen aanreiken bij een toe-passing of website.	8
2.3	Complexe leesfactoren van een wetenschappelijk artikel.	9
2.4	Prevalente leesgraadsscores.	10
2.5	Drie algemene technieken voor MTS.	13
2.6	Bewezen voordelen van MTS op mensen met dyslexie tijdens het be-grijpend lezen.	16
2.7	Beschikbare Nederlandstalige, Engelstalige en meertalige lexicale da-tabanken anno mei 2023.	20
2.8	De drie manieren om extraherende samenvatting mogelijk te maken volgens Verma en Verma (2020).	22
2.9	Overzicht van gekende voorleessoftware, tekstvereenvoudigings- en samenvattingstools die intuïtief zijn ontwikkeld voor de eindgebruiker (leerkracht of scholier).	26
2.10	HuggingFace beschikbare en ge-finetuned taalmodellen.	27
2.11	Tabel met alle GPT-3 parameters.	31
2.12	Technieken voor concrete en goed opgestelde prompts.	33
2.13	Samenvattend schema met vaak voorkomende struikelblokken bij NLP-toepassingen.	37
3.1	Shortlist van uit te testen tools en toepassingen voor tekstvereenvou-ding.	40
3.2	Richtlijnen waarop het onderzoek de toepassingen afvoert in de re-quirementsanalyse.	41
3.3	Bronvermeldingen voor de twee wetenschappelijke artikelen.	42
3.4	Het Moscow-schema voor de requirementsanalyse.	43
3.5	De toegepaste GPT-3-prompts in de requirementsanalyse.	44
3.6	Gebruikte taalmodellen in de vergelijkende studie	45
3.7	Gebruikte SpaCy word-embeddings	48
3.8	Meegegeven parameters bij HuggingFace-requests	48
3.9	De GPT-3-prompts die in de vergelijkende studie aan bod komen.	49
3.10	De objectieve metrieken voor de vergelijking van de vereenvoudigde teksten met de oorspronkelijke tekst en de referentieteksten.	52
3.11	Criteria voor menselijke observatie bij de vergelijkende studie.	54

3.12 Gebruikte programmeertalen in het prototype voor tekst vereenvoudiging.	55
3.13 Gebruikte Python-libraries en hun respectievelijke functie in het prototype.	57
3.14 Alle beschikbare functionaliteiten in	63
3.15 Tabel met de gebruikte prompts voor het lerarencomponent.	64
3.16 Gebruikte parameters om de definitie van een woord te genereren met GPT-3.	64
3.17 Benodigde labels voor een gepersonaliseerd document met Pandoc.	67
3.18 Beschikbare functionaliteiten in het scholierencomponent.	70
4.1 Afgetoetste criteria volgens de experimenten.	76
4.2 Aantal zinnen (gemeten met Spacy sentence embeddings) per tekst.	78

List of Listings

3.1 Script voor fase 1 van de vergelijkende studie.	46
3.2 Script voor de tweede fase van de vergelijkende studie.	47
3.3 Script voor de derde fase van de vergelijkende studie	49
3.4 Script voor fase 4 van de vergelijkende studie	52
3.5 De back end functie die de aanpassingen uit het formulier opslaat als sessievariabele.	57
3.6 De onload-functie die de gepersonaliseerde opmaakopties regelt bij het inladen van een webpagina.	58
3.7 Koppeling tussen front-end en back-end voor het inlezen van een wetenschappelijk artikel	59
3.8 Een PDF inlezen met PDFMiner	60
3.9 Een PDF inlezen met OCR	60
3.10 Het formatteren van de tekst naar een formaat voor de website.	61
3.11 Het doorlopen van de PDF-tekst op de webpagina en het toekennen van de span-tags.	62
3.12 Een functie om met GPT-3 een vereenvoudigde definitie voor een woord te genereren.	65
3.13 Een synoniem genereren of ophalen met GPT-3.	65
3.14 Een zin gepersonaliseerd vereenvoudigen met GPT-3.	66
3.15 Writer-klasse omvattende de code om dynamische PDF- en Word-documenten te genereren.	67
3.16 Een woordenlijst genereren met de Writer-klasse.	68
3.17 Een doorlopende tekst toevoegen aan het markdownbestand met de Writer-klasse.	68
3.18 Een opsomming toevoegen aan het markdownbestand met de Writer-klasse.	69
3.19 Een zip-bestand aanmaken met daarin een docx en pdf bestand van de vereenvoudigde tekst.	69
3.20 Een API-call sturen naar GPT-3 met een custom prompt.	71
3.21 Een woord aan de woordenlijst toevoegen in het scholierencomponent.	72
3.22 Zelfstandige naamwoorden in het scholierencomponent markeren.	73
D.1 Writer-klasse omvattende de code om dynamische PDF- en Word-documenten te genereren.	115
D.2 De toegepaste scripts voor het verwijderen van adjektieven en togglen van type woorden.	117

D.3 De toegepaste scripts voor enkel het scholierencomponent.	118
D.4 De toegepaste scripts voor enkel het lerarencomponent.	123
D.5 Dockerfile voor het prototype.	125
D.6 Script voor het opstarten van de Docker-container voor Windows-gebruikers	126
D.7 Script voor het opstarten van de Docker-container voor Unix-gebruikers	126

1

Inleiding

Lezen is een dagelijkse activiteit voor iedereen. Deze vaardigheid strekt zich uit tot elk aspect van het leven. Dit geldt des te meer in het onderwijs, waar leraren diverse leesmaterialen gebruiken om lesinhouden op een authentieke manier over te brengen. Scholen zetten wetenschappelijke artikelen in als leesvoer in de derde graad van het middelbaar onderwijs. De leesgraad van deze artikelen brengt echter een nieuwe uitdaging mee voor zowel scholieren als leerkrachten.

Zo lanceerde het Amerikaanse onderwijs het C.R.E.A.T.E.¹-initiatief. Het zet scholieren tussen 12 en 18 jaar aan om wetenschappelijke artikelen te lezen in plaats van enkel boeken. Ze hebben begrip van hoe wetenschappers onderzoek plannen, uitvoeren, en resultaten analyseren en interpreteren. Hoewel er geen vergelijkbare initiatieven bestaan in Vlaanderen, benadrukken de lerarenopleidingen het gebruik van diverse didactische leesmaterialen in de klas.

Het M-decreet en de leerplannen van het katholiek² en het gemeenschapsonderwijs³ adviseren de Vlaamse leerkrachten om hun lessen op een toegankelijke manier aanbieden. Zo nemen ze alle scholieren ongeacht leesachterstand mee in het verhaal.

Vlaanderen is met een jaarlijks budget van 32 miljoen een pionier in Europa op het gebied van artificiële intelligentie (AI) op de werkvloer (Crevits, 2022). Zo stampte de Vlaamse overheid verschillende projecten uit de grond om Vlaamse AI ontwikkelingen te ondersteunen en AI-softwarebedrijven te inspireren. Het amai!-project⁴

¹<https://teachcreate.org/>

²<https://pro.katholiekeonderwijs.vlaanderen/basisoptie-stem/ondersteunend-materiaal>

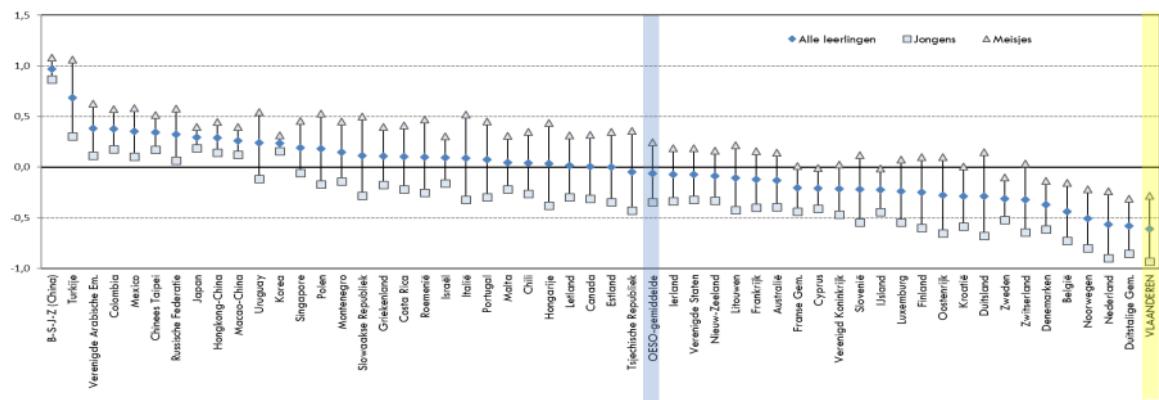
³<https://g-o.be/stem/>

⁴<https://amai.vlaanderen/>

brengt AI softwarebedrijven uit diverse domeinen samen, waaronder het onderwijs. Zij doelen op een automatisering van processen om de werkdruk bij leerkrachten te verminderen. Dit gebeurt door real-time ondertiteling in de klas en een taalassistent voor leerkrachten in meertalige klasgroepen.

1.1. Probleemstelling

Elke drie jaar nemen experts uit 79 geïndustrialiseerde landen de PISA-test af om de leesvaardigheid en wetenschappelijke geletterdheid van 15-jarige scholieren te meten. Uit de PISA-test van 2018 blijkt dat deze doelgroep in Vlaanderen zich echter negatief uit over leesplezier en daarmee de slechtst scorende doelgroep is van alle bevraagde landen. Zoals aangegeven in figuur 1.1, beschouwt bijna de helft van de bevraagden begrijpend lezen als tijdverspilling. Slechts 17% beschouwt lezen als een hobby. Dit is een dalende trend, want voordien lag deze trend hoger dan 20%. Boeiende topics uit vakliteratuur zoals Humo, Trends of EOS magazines kunnen belemmerd worden door deze vorm van media bij deze doelgroep.



Figuur (1.1)

Het leesplezier bij 15-jarigen volgens de PISA-test (De Meyer e.a., 2019).

Begrijpend lezen valt niet te omzeilen in onze huidige samenleving, maar het leesbegrip verschilt sterk onder studenten in het middelbaar onderwijs. Zo benadrukt de inspectie van Onderwijsinspectie Overheid Vlaanderen (2020) dat begrijpend lezen een essentiële vaardigheid, ook voor vakken buiten Nederlands. Bij wiskunde is begrijpend lezen van cruciaal belang bij complexe vraagstukken. Ook helpt begrijpend lezen studenten om STEM-vakjargon beter te begrijpen.

In het bijzonder hebben scholieren met dyslexie problemen met begrijpend lezen. Onderzoeken van Bonte (2020) en van der Meer (2022) schatten dat ongeveer 15% van de Vlaamse scholieren in het middelbaar onderwijs een vorm van dyslexie heeft. Scholieren met dyslexie ervaren moeite en hinder bij het lezen en spellen. On-

danks de bestaande ondersteuning blijven ze toch nog steeds de negatieve impact van hun leerstoornis ervaren. De gevolgen hiervan kunnen zich doorzetten na het middelbaar onderwijs (Lissens e.a., 2020). Leesvaardigheid blijft daarmee cruciaal voor succes op school en in het werkveld. Scholieren met dyslexie hebben moeilijkheden met deze vaardigheid, wat kan leiden tot onzekerheid en stress. Daarnaast zijn vooroordelen nog steeds een probleem en kunnen ze leiden tot stigmatisering. Echter toont onderzoek aan dat scholieren met dyslexie een sterke doorzettingsvermogen hebben en goede probleemplossers zijn (Bonte, 2020; Chesquière, 2018; Lissens e.a., 2020).

Het leerplan voor STEM-vakken stimuleert het gebruik van wetenschappelijke artikelen, maar houdt niet altijd rekening met de bijhorende complexe leesgraad. De ingewikkelde woordenschat en syntax in wetenschappelijke artikelen kunnen een hindernis vormen voor de begrijpelijkheid van een tekst. Voornamelijk scholieren met dyslexie kunnen de kerninhoud hierdoor moeilijk doorgronden. Handmatig vereenvoudigen van wetenschappelijke artikelen kan planning, tijd en energie van leerkrachten in het middelbaar onderwijs oplossen. Het Vlaamse onderwijsysteem staat onder druk en docenten hebben moeite om boven water te blijven.

AI-technologieën zijn vandaag voldoende hoogstaand om tekstvereenvoudiging te automatiseren en om een baanbrekende oplossing aan te bieden aan het middelbaar onderwijs. Het onderwijst past echter zelden soortgelijke technologieën toe. Er is terughoudendheid door enerzijds ouders van leerlingen (Martens e.a., 2021b), anderzijds door de weinige ontwikkelingen in schoolgerelateerde AI-software. Dit terwijl AI-ondersteuning in het onderwijs wel degelijk een positief effect heeft (Belpaeme e.a., 2018; Kraft, 2020).

Recente technologieën bieden reeds mogelijkheden tot tekstvereenvoudiging aan, maar zijn voorlopig enkel in commandline of met scripts ter beschikking. Voor het gebruik van taalmodellen of API's is uitgebreide informaticakennis nodig, waarover de meeste scholieren en leraren niet beschikken. Anderzijds zijn de huidige online tools te beperkt en eerder gericht op samenvatten; wat niet noodzakelijk bijdraagt tot een eenvoudigere tekst. Er is nood aan een intuïtieve en gebruikersvriendelijke toepassing die taalmodellen of API's kan integreren en aanpassen naargelang de specifieke behoeften van een student met dyslexie. Zo kan dit enerzijds ook de werkdruk bij leerkrachten verminderen, anderzijds scholieren in de derde graad ondersteunen bij het lezen van complexe wetenschappelijke artikelen.

1.2. Onderzoeksvraag

Dit onderzoek beschrijft het gebruik van artificiële intelligentie in de vorm van tekstvereenvoudiging, als advies voor implementatie in het onderwijs. Specifiek om

scholieren met dyslexie in de derde graad van het middelbaar onderwijs te ondersteunen bij het begrijpend lezen van wetenschappelijke artikelen. Hiervoor is de volgende onderzoeksvraag opgesteld:

- Hoe kan een wetenschappelijk artikel automatisch vereenvoudigd worden, gericht op de unieke noden van scholieren met dyslexie in de derde graad middelbaar onderwijs?

De oplossingen van volgende deelvragen kunnen een antwoord bieden op de onderzoeksvraag:

1. Welke specifieke noden hebben scholieren met dyslexie van de derde graad middelbaar onderwijs bij het begrijpen van complexere teksten? Aanvullend hierop:
 - Wat zijn de specifieke kenmerken van wetenschappelijke artikelen?
2. Welke aanpakken zijn er voor tekstvereenvoudiging?
 - Hoe verloopt de handmatige vereenvoudiging van teksten voor scholieren met dyslexie?
 - Welke toepassingen, tools en modellen zijn er beschikbaar om Nederlandse geautomatiseerde tekstvereenvoudiging met AI mogelijk te maken?
 - Hoe is de combinatie van geautomatiseerde tekstvereenvoudiging met gepersonaliseerde tekstvereenvoudiging mogelijk?
3. Welke functies ontbreken AI-toepassingen om geautomatiseerde tekstvereenvoudiging mogelijk te maken voor scholieren met dyslexie in de derde graad middelbaar onderwijs?
 - Welke manuele methoden voor tekstvereenvoudiging ontbreken in deze tools?
4. Met welke valkuilen bij taalverwerking met AI moeten ontwikkelaars rekening houden?
5. Welk taalmodel of LLM is geschikt voor de ATS van wetenschappelijke artikelen voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs, met dezelfde of gelijkaardige kwaliteiten als gepersonaliseerde MTS?
6. Wat zijn de nodige stappen bij de ontwikkeling van een intuitieve lokale webtoeassing die zowel scholieren met dyslexie als leerkrachten helpt?

1.3. Onderzoeksdoelstelling

Het onderzoek richt zich op het identificeren van technologische en logopedische aspecten voor AI-ontwikkelaars bij het creëren van een op maat gemaakte AI-toepassing voor geautomatiseerde tekstvereenvoudiging. Deze toepassing is specifiek ontwikkeld voor scholieren in de derde graad.

Het resultaat van het onderzoek is een prototype van een webtool voor tekstvereenvoudiging. De webtool heeft twee functies. Allereerst kan de tool wetenschappelijke artikelen vereenvoudigen op basis van de specifieke behoeften van scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Daarnaast biedt de tool een geautomatiseerde benadering voor lectoren om wetenschappelijke artikelen te vereenvoudigen met behulp van geselecteerde parameters. De webtool geeft de vereenvoudigde artikelen terug in een bruikbaar formaat (PDF of Word).

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

Hoofdstuk 2 geeft een overzicht van de stand van zaken binnen het onderzoeks-domein, op basis van een literatuurstudie.

Hoofdstuk 3 licht de methodologie toe en vermedt de gebruikte onderzoekstechnieken om een antwoord te kunnen formuleren op de onderzoeks-vragen. Eerst komt er een requirementsanalyse aan bod, gevolgd door de ontwikkeling van een prototype voor tekstvereenvoudiging. Hoofdstuk 4 bevat de resultaten van dit onderzoek.

Hoofdstuk 5 wordt de uiteindelijke conclusie en een antwoord op de onderzoeks-vragen gegeven. Ten slotte geeft Hoofdstuk 6 verdere aanbevelingen en aanzet voor toekomstig onderzoek binnen de bestudeerde domeinen.

2

Stand van zaken

2.1. Inleiding

Het onderzoek start met een uitgebreide literatuurstudie over de benodige kennis binnen het logopedisch, taal- en informatica vakdomein om geautomatiseerde en gepersonaliseerde vereenvoudigde teksten te verkrijgen van wetenschappelijke artikelen. Om een toepassing voor gepersonaliseerde en geautomatiseerde tekstvereenvoudiging van wetenschappelijke artikelen op maat van deze doelgroep aan te reiken, is het van cruciaal belang om de noden van scholieren met dyslexie in de derde graad van het middelbaar onderwijs te benoemen. Het onderzoek benoemt bewezen noden met behulp van een literatuurstudie. Daarnaast kaart het de huidige problemen bij wetenschappelijke artikelen aan. Wetenschappelijke artikelen vereenvoudigen op maat voor scholieren met dyslexie kan volgens taalexpertsen op verschillende manieren. Het is belangrijk om stil te staan bij de bestaande en reeds bewezen handmatige tekstvereenvoudigingstechnieken. Vervolgens komen technieken voor geautomatiseerde tekstvereenvoudiging (ATV) aan bod. Zowel de nodige informatie van taalverwerking met AI, als de huidige AI-technologieën voor tekstvereenvoudiging zijn gegeven. Ten slotte zijn AI-technologieën hoogstaand en ontwikkelaars maken deze alsmaar robuuster, maar het is cruciaal om bij dit onderzoek aandacht te besteden aan de mogelijke problemen die AI-ontwikkelaars moeten vermijden of waarvan zij zichzelf attent op moeten maken.

2.2. Specifieke noden en richtpunten

Om wetenschappelijke artikelen te vereenvoudigen op maat van de unieke noden van scholieren met dyslexie moet het onderzoek stilstaan bij de specifieke noden van scholieren met dyslexie in de derde graad van het middelbaar onderwijs, alsook bij de moeilijkheden bij het begrijpend lezen van wetenschappelijke arti-

kelen. Deze sectie bespreekt eerst algemeen hoe scholieren met dyslexie bij het begrijpend lezen kunnen worden geholpen. Daarna worden de belemmeringen en moeilijkheden van wetenschappelijke artikelen aangekaart. Deze sectie beantwoordt de volgende twee onderzoeks vragen:

- Welke specifieke noden hebben scholieren met dyslexie van de derde graad middelbaar onderwijs bij het begrijpen van complexere teksten?
- Wat zijn de specifieke kenmerken van wetenschappelijke artikelen?

2.2.1. Specifieke noden van scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

Leesvaardigheid is geen aangeboren vaardigheid, maar iets dat mensen zelf moeten aanleren (Bonte, 2020; van der Meer, 2022). Hoewel dit proces vlot kan verlopen, kunnen mensen met dyslexie benadeeld worden tijdens dit proces. Dyslexie wordt gekenmerkt door beperkt lezen en kan het voorlezen traag, radend en letter-voor-letter maken. Mensen met dyslexie kunnen tijdens het begrijpend lezen verschillende drempels ervaren; die worden in tabel 2.1 opgesomd.

Kenmerk	Bron
Trage woordbenoeming	(Bonte, 2020)
Problemen bij het leesbegrip	(Bonte, 2020; Gala & Ziegler, 2016)
Hardnekkig letter-voor-letter lezen	(Bonte, 2020; Zhang e.a., 2021)
Problemen met woordherkenning en -herinnering	(Bonte, 2020)
Moeite bij homofonische of pseudo-homofonische woordenschat	(Zhang e.a., 2021)
Moeite bij onregelmatige lettergreetcombinaties	Gala en Ziegler (2016)

Tabel 2.1: Specifieke drempels bij het begrijpend lezen van een tekst.

De digitalisering evolueerde de voorbije twintig jaar in stijgende lijn en scholieren in de tweede en derde graad zijn, door het gebruik van smartphones en laptops, hier het meeste vatbaar bij zowel thuisgebruik als gebruik in het onderwijs. Verder beschrijft dit artikel een checklist van technische elementen waaraan een webpagina of toepassing moet voldoen om een leesbare ervaring te voorzien voor scholieren met dyslexie. Tabel 2.2 geeft deze noden weer.

Kenmerk	Bron
Zachtgele, -groene -of bruine achtergrondkleur	Santana e.a. (2012)
Lettergrootte vergroten naar 14pt	(Rello & Baeza-Yates, 2015)
Woord- en karakterspatiëring	Rello en Baeza-Yates (2013) en Santana e.a. (2012)
Consistente lay-out	(Rello & Baeza-Yates, 2015)
Waarschuwingen geven omtrent formulieren, sessies (login)	
Consistente lay-out	(Botelho, 2021; Rello & Baeza-Yates, 2015)
Duidelijk zichtbare koppen- of heading-structuur	(Rello e.a., 2012a)
Duidelijke symbolen gebruiken	(Rello & Baeza-Yates, 2013)
Inhoud visueel groeperen	(Botelho, 2021; Rello & Baeza-Yates, 2015)
Huidige positie benadrukken	(Botelho, 2021)

Tabel 2.2: Oplossingen die software-ontwikkelaars kunnen aanreiken bij een toepassing of website.

2.2.2. Specifieke kenmerken van wetenschappelijke artikelen

Wetenschappelijke artikelen zijn van cruciaal belang voor het verspreiden van nieuwe kennis en onderzoeksresultaten, maar toch blijven ze voor velen een mysterieus en ontoegankelijk gebied vanwege de complexiteit van de inhoud en het technische jargon dat onmisbaar lijkt te zijn (Ball, 2017). Dit kan het begrip van de artikelen bemoeilijken, vooral bij begrijpend lezen, en vormt daarmee een extra obstakel bij het gebruik van wetenschappelijke artikelen als bron van kennis tijdens de les. Wetenschappelijke artikelen volgen IMRAD, een uniform formaat voor gepubliceerde wetenschappelijke artikelen, dat bestaat uit vijf hoofdstukken: samenvatting, inleiding, methodologie, resultaten en discussie. Hoewel het middelbaar en hoger onderwijs deze artikelen gebruiken als leermiddel, is de inhoud van een hoger niveau en voornamelijk gericht op mensen uit het vakgebied. Charlesworth Author Services (2021) en Pain (2016) benadrukken de complexiteit van wetenschappelijke artikelen en de volgende aspecten waarom ze moeilijk te interpreteren zijn.

Tabel 2.3 somt deze factoren op. Hoewel wetenschappelijke artikelen over een grote drempel bezitten, betrekken ze jongeren met wetenschappelijk onderzoek,

alsook leren ze een discussieerbaar en kritische vaardigheid.

Probleem	Oplossing
Veel informatie in een compact formaat of <i>high information density</i>	Extra uitleg schrijven bij compacte zinnen, zoals extra uitleg bij woorden of zinnen herschrijven.
Hoog gebruik van meerlettergrepige woorden	Eenvoudigere synoniemen gebruiken.
Wetenschappelijk jargon	Rekening houden met een doelgroep buiten het vakgebied door eenvoudigere synoniemen te schrijven. Indien deze niet beschikbaar zijn, kan er extra uitleg als alternatief worden gegeven.
Complexe concepten	Paragrafen herschrijven zodat ze eerst uitleg geven op een high-level niveau. Vervolgens lagen van complexiteit toevoegen om de lezer te begeleiden doorheen de methodologie, discussie en conclusie van het wetenschappelijk artikel.
Cijfermateriaal bij resultaten	De interpretatie van percentages of cijfermateriaal schrijven. Zoals 'ongeveer een kwart van de bevolking' in plaats van '24.97% van de bevolking'.

Tabel 2.3: Complexe leesfactoren van een wetenschappelijk artikel.

Scholieren met verschillende achtergronden verschillen vaak in achtergrondkennis, wat invloed kan hebben op het tekstbegrip (De Meyer e.a., 2019). Zo kunnen scholieren met een achtergrond in fysica sneller de draad oppikken bij het lezen van fysica-gerelateerde artikelen dan scholieren met een economische achtergrond. Dit maakt het moeilijk om de leesbaarheid van een tekst objectief te beoordelen. Het jargon kan voor de ene groep scholieren makkelijk zijn, en voor de andere groep moeilijk.

Onderzoeken en ontwikkelaars zorgden voor de geautomatiseerde berekening van leesmetrieken. Dankzij Python-libraries, zoals Textstat¹ en Readability², kunnen ontwikkelaars via commandline-toepassingen deze processen om leesmetrieken te berekenen automatiseren. Zo zijn drie prevalent leesbaarheidsscores weerge-

¹textstat

²<https://pypi.org/project/readability/>

geven in tabel 2.4. Daarnaast kunnen toepassingen, zoals Textinspector³, analytisch inzicht geven in de complexiteit van Engelstalige teksten. Er bestaat echter geen alternatief dat leesgraadsscores kan berekenen voor Nederlandstalige wetenschappelijke artikelen. Het is wel belangrijk om te benadrukken dat deze leesbaarheidsscores geen rekening houden met de achtergrondkennis van mogelijke lezers.

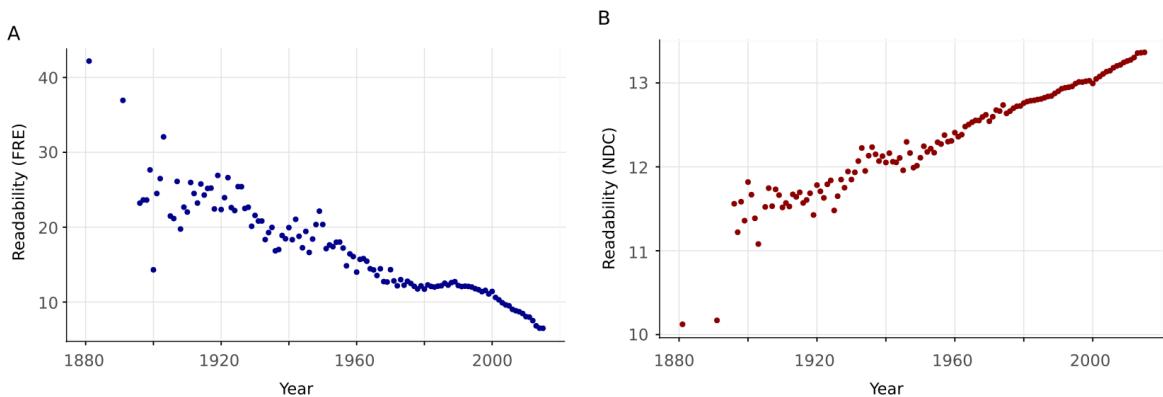
Score	Uitleg
Flesch Reading Ease (FRE)	Deze leesbaarheidsscore berekent de moeilijkheidsgraad op zinsbasis. Hoe hoger de score, hoe 'eenvoudiger' de zin.
Gunning FOG (FOG)	In tegenstelling tot FRE, berekent FOG de moeilijkheidsgraad op basis van de volledige tekst.
Complexe woorden volgens <i>Dale-Chall Index</i> (DCI)	Deze lijst omvat woorden die experimenten bij Amerikaanse tieners als complex omschrijven. De DCI werkt per leeftijdscategorie.

Tabel 2.4: Prevalente leesgraadsscores.

Diverse onderzoeken van de afgelopen tien jaar wijzen uit dat wetenschappelijke teksten steeds complexer worden. Dat maakt deze teksten voor niet-experten en niet-doctoraatsstudenten minder toegankelijk, vanwege het gebruik van technisch jargon en ingewikkelde zinsstructuren (Ball, 2017; Jones e.a., 2019; Plavén-Sigray e.a., 2017). Deze trend begon volgens onderzoek al in de tweede helft van de twintigste eeuw (Hayes, 1992).

Volgens onderzoek van Plavén-Sigray e.a. (2017) is de taal van wetenschappelijke artikelen de laatste jaren steeds complexer geworden. Uit een vergelijkende studie tussen abstracts en de rest van de inhoud van wetenschappelijke tijdschriften blijkt dat abstracts het meest complexe deel van een artikel vormen. De evolutie van de leesbaarheid wordt weergegeven in figuur 2.1, dat ook de FRE- en NDC-scores toont. Het onderzoek schat dat 22% van alle wetenschappelijke artikelen op het niveau van een masterstudent in het Engels geschreven zijn, tegenover 14% in 1960. Deze trend is belangrijk om op te volgen in de komende decennia, omdat het een obstakel kan vormen voor toekomstige generaties.

³<https://textinspector.com/>

**Figuur (2.1)**

De toename van vereiste leesgraad volgens FRE (links) en NDC (rechts). Bron: (Plavén-Sigray e.a., 2017)

Onbegrijpelijke en ontoegankelijke zinsstructuren hinderen ook vakexperten. Zo toonde onderzoek van McNutt (2014) aan dat begrip van de methodologie en resultaten cruciaal is in het kader van reproduceerbaarheid; enkel zo kunnen wetenschappers op correcte wijze een studie reproduceren en wetenschappelijke inzichten bevestigen of met verdere resultaten verrijken. Experimenten van Hubbard en Dunbar (2017) wijzen namelijk uit dat het net vooral de methodologie en resultaten van een wetenschappelijk artikel zijn die een hoge leesgraad vergen. In deze context is ook het onderzoek van Hartley (1999) en Snow (2010) relevant waarin ze aantonen dat het herschrijven van abstracts de begrijpbaarheid ervan kan verhogen.

Volgens Hollenkamp (2020) en McCombes (2022) moeten vereenvoudigde of samengevatte wetenschappelijke artikelen drie vragen kunnen beantwoorden: waarom werd het onderzoek uitgevoerd, wat zijn de experimenten en wat zijn de conclusies van de onderzoekers? Dit omvat de achtergrondinformatie, hypotheses, methoden, resultaten, implicaties, beperkingen en aanbevelingen. De tekst omzetten in een ander formaat zoals post-it notes, tabelvorm of opsommingen, maakt het beter begrijpbaar (Rijkhoff, 2022).

Wetenschappelijke artikelen zijn voornamelijk in PDF-formaat terug te vinden. Dit formaat valt eenvoudig in te lezen met Python-pakketten, waaronder PDFMiner of PyPDF. Wel ondervinden ontwikkelaars soms problemen bij het in-lezen van PDF-bestanden: niet alle tekst of kan uit de PDF worden geëxtraheerd. Als oplossing kunnen ontwikkelaars gebruik maken van *optical character recognition* of OCR. In Python zijn er pakketten die deze technologie eenvoudig kunnen realiseren, namelijk EasyOCR en Tesserat.

2.3. Aanpakken voor tekstvereenvoudiging

2.3.1. Manuele tekstvereenvoudiging

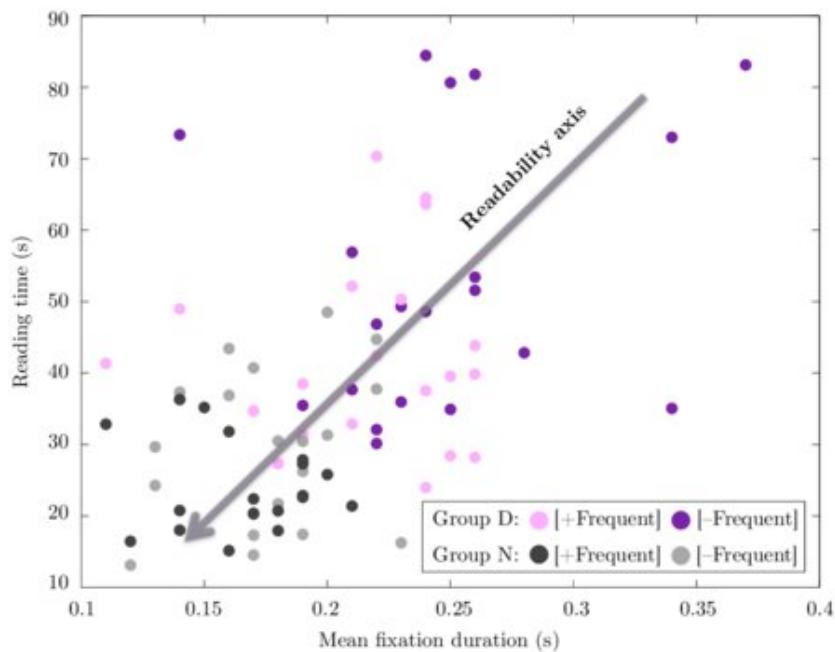
Voor sommige lezers kan het begrijpend lezen van een complexe tekst echter een uitdaging zijn, zoals scholieren met dyslexie. *Manual tekst simplification* of MTS kan deze groep helpen (Siddharthan, 2014). De techniek van MTS gebruikt eenvoudige woordenschat en zinsstructuren en maakt structurele aanpassingen om de tekst vlotter leesbaar te maken. MTS is het proces dat het technische leesniveau en het woordgebruik van een geschreven tekst vermindert. Dit leidt tot een betere leeservaring zonder het verlies van de kerninhoud tijdens het lezen van de tekst.

Type vereenvoudiging	Techniek	Bron
LS	Moeilijke woorden vervangen door eenvoudigere synoniemen Woorden- en synoniemenlijst maken Dubbelzinnige woorden vervangen Idiomen vervangen Rekening houden met het gekende jargon van de doelgroep	(Rello, Baeza-Yates & Sag-gion, 2013; Siddharthan, 2014)
SS	Tangconstructies aanpassen Zinnen langer dan negen woorden inkorten Verwijswoorden aanpassen Voorzetseluitdrukkingen aanpassen Samengestelde werkwoorden aanpassen Actieve stem gebruiken Onregelmatige werkwoorden vermijden	(Ruelas In-zunza, 2020)
Structurele aanpassingen	Marges aanpassen Lettertype en -grootte aanpassen Woord- en karakterspatiëring aanpassen Herschrijven en -structureren als opsomming of tabelvorm	

Tabel 2.5: Drie algemene technieken voor MTS.

2.3.2. Bevoordelende effecten van MTS bij scholieren met dyslexie

Onderzoek toont aan dat vereenvoudigde teksten het leesbegrip en woordherkenning van kinderen met dyslexie significant kunnen verbeteren (Rivero-Contreras e.a., 2021). Bovendien blijkt uit experimenten dat frequent woordgebruik de ontcijfertijd bij mensen met dyslexie significant vermindert, en dat teksten met verminderde lexicaal complexiteit minder leesfouten opleveren voor mensen met dyslexie



Figuur (2.2)

De gemeten *mean fixation duration* tijdens begrijpend lezen uit het onderzoek van Rello, Baeza-Yates, Dempere-Marco e.a. (2013).

(Gala & Ziegler, 2016; Rello, Baeza-Yates, Dempere-Marco e.a., 2013). Deze studies benadrukken ook moeilijkheden van kinderen met dyslexie bij het lezen van woorden met onregelmatige lettergreepcombinaties. Mensen zonder dyslexie bereiken doelwaarden onder optimale omstandigheden, zoals aangegeven door de richting van de pijl op figuur 2.2. Het gebruik van veelvoorkomende woorden vermindert de decodeertijd en verbetert het leesbegrip voor mensen met dyslexie.

Hoewel onderzoeken de positieve effecten van lexicale vereenvoudiging voor lezers met dyslexie onderstrepen, is er relatief weinig onderzoek gedaan naar de effecten van syntactische vereenvoudiging op kinderen en scholieren met dyslexie. In het experiment van Linderholm e.a. (2000) had het aanpassen van causale structuren een significant effect op het leesbegrip en de foutenmarge van de bevraagden met een lage leesgraad. Door coherentieonderbrekingen te herstellen en tekstgebeurtenissen in een tijdsafhankelijke volgorde te plaatsen, konden zowel vaardige als minder vaardige lezers profiteren van de revisies. Verbaal parafraseren had geen significant effect op lezers met dyslexie, volgens Rello, Baeza-Yates en Saggion (2013). De bevraagden waren tussen de 13 en 37 jaar oud, met een gemiddelde leeftijd van 21 jaar. Het tekstformaat bleef ongewijzigd, maar lettertypes werden aangepast.

Het onderzoek van Nandhini en Balasundaram (2013) experimenteerde met een an-

dere vorm van aanpassingen om de leesbaarheid van teksten te verhogen, namelijk gepersonaliseerde samenvattingen. Het experiment in het onderzoek maakt gebruik van onaangepaste zinnen uit de oorspronkelijke tekst die op maat van de lezer zijn gepresenteerd en herstructureert deze volgens de oorspronkelijke tekst. Door de belangrijkste zinnen onaangepast te laten en de structuur aan te passen, is de tekst toegankelijker voor de lezer. Hoewel de onderzoekers de resulterende logische structuur in twijfel trokken, was de leesbaarheid van teksten bij de deelnemers significant beter dan bij de oorspronkelijke tekst, zonder negatieve effecten op het leesbegrip.

Tot slot hebben onderzoeken aangetoond dat scholieren met dyslexie gevoeliger zijn voor veranderingen in visuele parameters, zoals lettertype, karakterafstand, tekst- en achtergrondkleur en grijswaarden. Minimalistische ontwerpen met pictogrammen behoren tot de aanbeveling om de leesbaarheid te verbeteren, evanals lettergrootte groter dan 14pt en een *sans-serif*, *monospaced* of *roman* lettertype. Volgens Bezem en Lugthart (2016), Rello en Baeza-Yates (2015) en Rello en Bigham (2017) zijn lichtgrijze achtergronden met zwart lettertype op een gele achtergrond, of zachtgele, -groene of lichtblauwe achtergrondkleuren de beste kleurencombinaties. Het gebruik van lettertypen zoals OpenDys heeft geen effect op lezers met of zonder dyslexie, terwijl cursieve lettertypen worden afgeraden, aldus Rello en Baeza-Yates (2013) en Rello en Baeza-Yates (2015).

Tabel 2.6 somt de bewezen strategieën op samen met de bewezen voordelen tijdens het begrijpend lezen.

Techniek	Bewezen voordeel	Bron
Frequent woordgebruik	Lagere decodeertijd Beter leesbegrip	
Verwerpen van onregelmatige lettergrepen	Verminderde decodeertijd Beter leesbegrip	
Causale structuren aanpassen	Beter leesbegrip Minder fouten bij het begrijpend lezen	
Tekstgebeurtenissen in een tijdsafhankelijke volgorde plaatsen	Beter leesbegrip bij het reviseren	
Coherentieonderbrekingen herstellen	Beter leesbegrip bij het reviseren	
Gepersonaliseerde samenvatting	Betere leesbaarheid	
Zachtkleurige achtergrond	Betere leesbaarheid	
Niet-cursieve, sans-serif lettertypen	Betere leesbaarheid	
Lettertype groter dan 14pt	Betere leesbaarheid	

Tabel 2.6: Bewezen voordelen van MTS op mensen met dyslexie tijdens het begrijpend lezen.

2.3.3. Aanpak voor ATS.

De laatste evolutie in machinaal leren biedt een mogelijkheid om dit proces te automatiseren. *Automatic text simplification* of ATS is een onderdeel van natuurlijke taalverwerking binnen machinaal leren (ML). Dit omvat technieken zoals tekstanalyse, taalherkenning, -generatie, spraakherkenning en -synthese, en semantische analyse. NLP stelt computers in staat om menselijke taal te begrijpen en te communiceren op een natuurlijke manier. De begrippen die volgen worden behandeld in Eisenstein (2019) en Sohom (2019) en zijn cruciaal voor de daaropvolgende concepten.

Zo dient tokenisatie om zinnen op basis van tokens te splitsen. Er zijn vier manieren om tokens in een tekst te splitsen en zo een woordenschat op te bouwen, namelijk

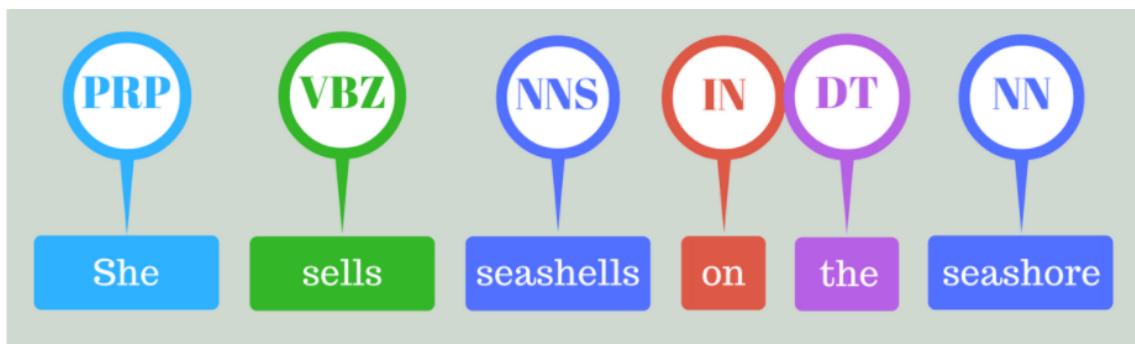
op woord-, karakter-, subwoord- en zinniveau, volgens onderzoek van Menzli (2023). Bij karaktertokenisatie neemt de inputlengte toe, maar dit heeft volgens Ribeiro e.a. (2018) weinig bruikbare *use cases*. Het opsplitsen van zeldzame woorden in kleinere stukken om een woordenschat op te bouwen biedt voordelen ten opzichte van woordtokenisatie, aldus (Iredale, 2022).

In NLP is het lemmatiseren gebaseerd op *stemming*, ofwel een NLP-taak waarbij de stam van een woord wordt genomen, maar het houdt ook rekening met de betekenis van elk woord. Er zijn Nederlandstalige modellen beschikbaar voor lemmatiseren, zoals JohnSnow⁴. Bij omgekeerd lemmatiseren wordt een afgeleide van de stam bepaald, bijvoorbeeld enkelvoud of meervoud voor zelfstandige naamwoorden zoals 'hond' (Eisenstein, 2019). Bij het *parsen* krijgt elk woord of zinsdeel een label toegewezen, zoals zelfstandig naamwoord, bijwoord, werkwoord, bijzin of stopwoord. Het identificeren van zinsdelen wordt *chunking* genoemd. *Parsing* is vatbaar op ambiguïteit, omdat bijvoorbeeld een 'plant' niet gelijk is aan de vervoeging van het werkwoord 'planten' (Eisenstein, 2019).

In NLP baseert het lemmatiseren zich op stemming, een NLP-taak waarbij deze de stam van een woord neemt, maar ook rekening houdt met de betekenis van elk woord. Er zijn Nederlandstalige modellen beschikbaar voor lemmatiseren, zoals JohnSnow⁴. Omgekeerd lemmatiseren werkt door een afgeleide vorm van de stam te bepalen, bijvoorbeeld enkelvoud of meervoud voor zelfstandige naamwoorden als 'hond' (Eisenstein, 2019). Bij het *parsen* krijgt elk woord of zinsdeel een label toegewezen, zoals zelfstandig naamwoord, bijwoord, werkwoord, bijzin of stopwoord. De identificatie van zinsdelen heet *chunking*. *Parsing* is vatbaar op ambiguïteit omdat bijvoorbeeld 'een plant' niet gelijk is aan de vervoeging van het werkwoord 'planten' (Eisenstein, 2019).

Om de betekenis van elk woord in een tekst te begrijpen, moet een machine in staat zijn om de betekenis achter elk token te begrijpen. Dit is waar *sequence labeling* om de hoek komt kijken, volgens Eisenstein (2019). Elk woord in een tekst krijgt een label voor *Part-of-Speech* (PoS) of *Named-Entity-Recognition* (NER). Deze fase van NLP achterhaalt de structuur van een tekst. PoS-tagging richt zich op de grammaticale categorieën van woorden, terwijl NER-labeling zich richt op het herkennen van specifieke entiteiten in een tekst. Bij PoS-tagging worden de woorden in een zin geanalyseerd. Elk woord krijgt een koppeling aan een grammaticale categorie, zoals een zelfstandig naamwoord, werkwoord, bijvoeglijk naamwoord of bijwoord. PoS-tagging helpt bij het begrijpen van de syntactische structuur van een zin en is nuttig bij parsing en machinevertaling. Een voorbeeld van PoS-tagging is te zien in figuur 2.3 en is afkomstig uit Bilici (2021).

⁴https://nlp.johnsnowlabs.com/2020/05/03/lemma_nl.html

**Figuur (2.3)**

Voorbeeld van PoS-labeling uit het artikel van Bilici (2021).

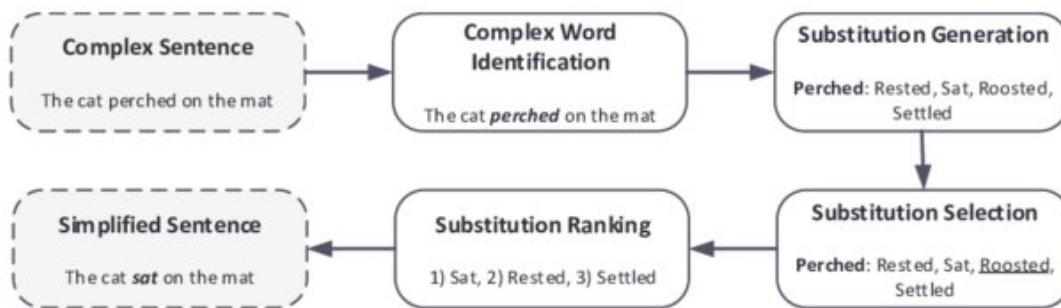
NER-labeling is een techniek om namen van personen, organisaties en locaties te herkennen en te classificeren. Het haalt specifieke informatie uit een tekst, zoals het identificeren van de namen van personen, plaatsen of bedrijven die in nieuwsartikelen, of het extraheren van belangrijke data of getallen uit financiële rapporten, aldus Jurafsky e.a. (2014). Er zijn vier vormen van NER-labeling, zoals beschreven door J. Li e.a. (2018): *dictionary-based*, *rule-based*, *ML-based* en *deep learning-based*. De eerste twee maken gebruik van vooraf gedefinieerde woordenboeken en regels, terwijl de laatste twee gebruik maken van statistische of neurale netwerken om te leren hoe entiteiten te herkennen. Elke vorm maakt gebruik van representaties om entiteiten te modelleren. Poel e.a. (2008) heeft een neurale netwerkmodel onderzocht voor PoS-tagging van Nederlandstalige teksten. Het model behaalde een nauwkeurigheid van 97,88% voor bekende woorden en 41,67% voor onbekende woorden en maakte gebruik van de Corpus Gesproken Nederlands (CGN) als trainingsdata. In de verwerking van tekst maken NLP-systemen gebruik van embeddings om woorden numeriek te representeren. Traditionele word embeddings bouwen een woordenschat op zonder de betekenis ervan in context te begrijpen. Contextuele word embeddings begrijpen daarentegen wel de context van een woord (Eisenstein, 2019).

2.4. De verschillende soorten ATS

Tekst vereenvoudiging kan bijdragen aan het begrijpen van complexe informatie. Zoals onderzocht door Siddharthan (2014), zijn er vier soorten transformaties bij geautomatiseerde tekst vereenvoudiging, waaronder lexicale vereenvoudiging, waarbij eenvoudigere synoniemen de complexe woorden vervangen. Dit heet lexical simplification (LS) of lexicale vereenvoudiging. Bijvoorbeeld, 'klevend' kan 'adhesief' vervangen. Kandula e.a. (2010) noemt twee manieren om lexicale vereenvoudiging te bewerkstelligen: het vervangen van het woord door een synoniem of het genereren van extra uitleg. De zinsstructuur blijft hetzelfde en de kerninhoud en

benadrukking van de tekst blijven behouden. Het doel van lexicale vereenvoudiging is om de moeilijkheidsgraad van de woordenschat in een zin of tekst te verlagen.

Diverse onderzoeken hebben aangetoond dat lexicale vereenvoudiging een belangrijke bijdrage kan leveren aan het begrijpen van complexe informatie, en in dit kader wordt de pipeline zoals weergegeven in figuur 2.4 vaak gebruikt, bijvoorbeeld in onderzoeken van Bingel e.a. (2018), Bulté e.a. (2018) en Paetzold en Specia (2016). Deze pipeline omvat bij de vermelde onderzoeken telkens minstens vier stappen, waarbij de eerste stap *Complex Word Identification* (CWI) is, een gesuperviseerde NLP-taak die moeilijke woorden of *multi-word expressions* (MWE) in een tekst identificeert (Gooding & Kochmar, 2019; Shardlow, 2013). De LS, waarbij eenvoudigere synoniemen de moeilijkere woorden vervangen, komt na CWI. Hier kunnen ook verklarende beelden of definities komen (Kandula e.a., 2010; Zeng e.a., 2005). Een goede uitvoering is van cruciaal belang bij CWI, omdat een lage recall van dit component zal resulteren in een uitvoertekst zonder vereenvoudiging van moeilijke woorden zoals opgemerkt door Shardlow (2013). Er zijn verschillende manieren geïdentificeerd om substitutiegeneratie uit te voeren, zoals opgesomd in tabel 2.7. Recent onderzoek, zoals dat van Zhou e.a. (2019), gebruikt ook een extra *Substitution Ranking* (SR) stap om substituties te rangschikken op basis van relevantie.



Figuur (2.4)

Een pipeline voor LS volgens Althunayyan en Azmi (2021).

Databank	Ondersteunde talen
Engels	WordNet SWORDS LSBert
Nederlands	Celex NT2Lex Cornetto
Meertalig (Engels, Duits, Spaans en Portugees)	PHOR-in-One

Tabel 2.7: Beschikbare Nederlandstalige, Engelstalige en meertalige lexicale databanken anno mei 2023.

Syntactische vereenvoudiging is een toepassing om de complexiteit van een zin te verminderen. Het past de grammatica en zinsstructuur van de tekst aan. Dit gebeurt door het combineren van twee zinnen tot één eenvoudigere zin of door de syntax te vereenvoudigen, terwijl de inhouden behouden blijft. Een voorbeeld van zo'n model is ontwikkeld door Kandula e.a. (2010) voor medische informatie. Het model bestaat uit drie modules, die zinnen met meer dan tien woorden vereenvoudigen en eventueel vervangen door kortere zinnen. Het omvat een PoS Tagger, een Grammar Simplifier en een Output Validator als onderdelen van de architectuur.

1. Ten eerste wordt de PoS Tagger-functie uit het open-source pakket OpenNLP⁵ gebruikt.
2. Vervolgens splitst de *Grammar Simplifier*-module lange zinnen in kortere zinnen door middel van het identificeren van POS-patronen en het toepassen van transformatieregels.
3. Tot slot controleert de *Output Validator*-module de grammatica en leesbaarheid van de output van de Grammar Simplifier.

Geautomatiseerde tekstvereenvoudiging is geen nieuw concept. Volgens onderzoeken van Canning e.a. (2000) en Siddharthan (2006) waren de eerste aanpakken op geautomatiseerde tekstvereenvoudiging gebouwd op rule-based modellen. Deze modellen bewerken de syntax door zinnen te splitsen, te verwijderen of

⁵<https://opennlp.apache.org/>

de volgorde van de zinnen in een tekst aan te passen. Lexicale vereenvoudiging kwam hier niet aan de pas. Enkel bij recentere onderzoeken van Bulté e.a. (2018) en Coster en Kauchak (2011) werd het duidelijk hoe lexicale en syntactische vereenvoudiging gecombineerd kon worden.

Vroegere onderzoeken tonen aan dat geautomatiseerde tekstvereenvoudiging al geruime tijd bestaat. Zo hebben Canning e.a. (2000) en Siddharthan (2006) onderzocht dat de eerste methoden gebaseerd waren op *rule-based* modellen die de syntaxis van de tekst bewerken door zinnen te splitsen, te verwijderen of te herschikken. Lexicale vereenvoudiging speelde hierbij geen rol. Enkel de recentere onderzoeken van Pas bij recentere onderzoeken, zoals die van Bulté e.a. (2018) en Coster en Kauchak (2011) tonen de mogelijkheid aan om lexicale en syntactische vereenvoudiging te combineren.

Om wetenschappelijke artikelen toegankelijker en begrijpelijker te maken, is het van belang om de kernpunten van een artikel op een duidelijke en beknopte manier samen te vatten. Hoewel samenvatten niet gericht is op het vereenvoudigen van de tekst, is het wel een techniek die noodzakelijk is om de semantiek achter een tekst met zo min mogelijk woord- of tekens te kunnen begrijpen. Full-text-search en gepersonaliseerde informatiefiltering benadrukken het belang van deze op maat gemaakte samenvattingen. Een samenvattingssysteem bestaat uit drie fases: analyse van de brontekst, identificatie van de kernpunten en samenvoeging van deze kernpunten tot één overzichtelijke tekst. Het machinaal samenvatten van teksten kan op twee manieren: door extractie en door abstractie (DuBay, 2004; Hahn & Mani, 2000).

Het proces van extraherend samenvatten markeert de belangrijkste zinnen in een tekst en herschrijft ze. Dit kan echter leiden tot onsamenhangende uitvoertekst, zoals Khan (2014) heeft aangetoond. Er zijn verschillende methoden beschikbaar om de kernzinnen te bepalen, zoals woordfrequentie, zinpositie -en gelijkenissen, de cue-methode, titels, *proper nouns*, woordgebruik en de afstand tussen *text unit entities*, aldus Khan (2014). Verschillende technieken voor het extraherend samenvatten van teksten, waaronder graafgebaseerde methoden, maximal marginal relevance (MMR) en metaheuristiek gebaseerde ES, zijn onderzocht door Verma en Verma (2020) en worden verder beschreven in tabel 2.8.

MMR-gebaseerde ES	Deze traditionele techniek gebruikt de maximaal marginale relevantiescore (MMR) om de relevantie en diversiteit van gemarkeerde zinnen te bepalen. Zorgt ervoor dat de geselecteerde zinnen niet te veel overlappen in inhoud en relevantie. Kan leiden tot betere samenvattingen, maar vereist meer rekenkracht en tijd.
Graafgebaseerde ES	Deze techniek vertegenwoordigt een document als een graaf van zinnen en gebruikt algoritmen om de belangrijkste zinnen te bepalen en redundantie te vermijden. Kan zowel voor lange wetenschappelijke artikelen als korte nieuwsartikelen goede resultaten opleveren (Lin & Bilmes, 2010; McDonald, 2007).
Metaheuristiek-gebaseerde ES	Deze techniek maakt gebruik van optimalisatie-algoritmen zoals genetische algoritmen en zweromoptimalisatie om de belangrijkste zinnen in een tekst te vinden (Premjith e.a., 2015; Verma & Verma, 2020). Evaluatiefunctie kan in een lokaal optimum vastlopen afhankelijk van de gebruikte criteria. (Rani & Kaur, 2021).

Tabel 2.8: De drie manieren om extraherende samenvatting mogelijk te maken volgens Verma en Verma (2020).

Vooroordelen of een *bias* kan de extraherende samenvatting van nieuwsartikelen beïnvloeden, zo blijkt uit experimenten uitgevoerd door McKeown e.a. (1999). Deze vorm van samenvatten neemt de zinnen over zoals ze zijn. Hahn en Mani (2000) bouwde verder op deze experimenten door het combineren van *knowledge-rich* en *knowledge-poor* methoden, wat resulterde in significante verbeteringen. Bij het extraherend samenvatten is het van belang om de meest relevante tekstgedeeltes te selecteren, meestal in de vorm van zinnen. Om de lexicale en statistische relevantie van een zin te kunnen bepalen, noemen Hahn en Mani (2000) twee methoden:

- Het lineaire gewichtsmodel, waarbij elke teksteenheid wordt gewogen op basis van factoren zoals de positie van de zin en het aantal keren dat deze voorkomt.
- Het gewichtsmodel op basis van de statistische relevantie van een eenheid, waarbij rekening wordt gehouden met de aanwezigheid van woorden in titels.

Om de nauwkeurigheid van modellen te verbeteren, ontwikkelden Nallapati e.a. (2017) *SummaRuNNer*. Dit model kan teksten extraherend samenvatten door een neuraal netwerk. Zo gebruikt het *PyTorch* en bestaat het uit drie modellen: een recurrent neuraal netwerk, een convolutioneel recurrent neuraal netwerk en een *hiérarchical attention network*.

Extraherende samenvattingen kunnen leiden tot een onsaamenhangende tekst. Abstraherende samenvatting kan een oplossing bieden, omdat het rekening houdt met de samenhang van een tekst. Onderzoek van Gupta en Gupta (2019) wijst twee benaderingen voor abstraherende samenvatting: semantisch-gebaseerd en structuurgebaseerd. De structuurgebaseerde methode gebruikt regels om belangrijke informatie in de tekst te vinden, maar dit kan leiden tot samengevattede zinnen van lage taalkundige kwaliteit en grammaticale fouten. De semantisch-gebaseerde benadering daarentegen gebruikt de betekenis van de tekst om korte en duidelijke samenvattingen te maken met minder redundante zinnen en een betere taalkundige kwaliteit. Een extra parsingfase kan van pas komen volgens de onderzoeken. Onderzoeken van Cao (2022) en Suleiman en Awajan (2020) wijzen *deep learning*-methoden uit om automatisch abstraherende samenvattingen te genereren. Zo kunnen RNN's, CNN's en Seq2Seq dienen om abstraherende samenvatting mogelijk te maken.

Ontwikkelaars moeten een andere aanpak gebruiken wanneer zij een systeem willen ontwikkelen voor *long text summarization* of LTS, zoals bij boeken of wetenschappelijke tijdschriften. Zo kan het opsplitsen van de tekst leiden tot het breken van saamenhangende paragrafen. Dat kan later resulteren in redundante tekst in het samengevatte document. Zo raadden onderzoeken van Hsu e.a. (2018) en Huang e.a. (2019) aan om zowel extraherend als abstraherende samenvatting toe te passen. Om deze reden zou een *hybrid summarization pipeline* twee fasen moeten bevatten: een inhoudselectiefase waarbij het systeem kernzinnen extraheert, gevolgd door een parafraserende fase.

Daarnaast moeten ontwikkelaars ook rekening houden met de doelgroep wanneer zij een systeem of model uitkiezen voor tekstvereenvoudiging of samenvatting. Zo moeten ontwikkelaars rekening houden met de individuele behoeften en uitdagingen van elke scholier, volgens Gooding (2022). Dyslexie kan zich namelijk op verschillende manieren uiten bij verschillende scholieren, waarbij bijkomende symptomen niet altijd van invloed zijn op de spellingprestaties van een scholier. Om deze reden is het belangrijk om een toepassing te ontwerpen die rekening houdt met de diversiteit van dyslexie.

2.5. Beschikbare tools en taalmodellen

Het kan moeilijk zijn voor scholieren met dyslexie om goed te lezen en te schrijven. Gelukkig zijn er verschillende softwareprogramma's en tools beschikbaar om hen te ondersteunen. Deze sectie gaat in op de nationale en internationale software die specifiek is ontworpen voor scholieren met dyslexie om hen te helpen bij het lezen van teksten. Er zal voornamelijk worden gekeken naar beschikbare software in Vlaamse middelbare scholen, chatbots zoals Bing Chat en ChatGPT, en software die speciaal is ontwikkeld om dyslexie bij het lezen te ondersteunen. Deze sectie beantwoordt de volgende deelvraag:

- Welke toepassingen, tools en modellen zijn er beschikbaar om Nederlandstalige geademtiseerde tekstvereenvoudiging met AI mogelijk te maken?

Scholieren met dyslexie krijgen in het middelbaar onderwijs enkel ondersteuning in de vorm van voorleessoftware (De Craemer e.a., 2018; OnderwijsVlaanderen, 2023). Het ministerie van Onderwijs in Vlaanderen biedt licenties aan voor verschillende softwarepakketten zoals SprintPlus, Kurzweil3000, Alinea Suite, IntoWords en TextAid, die scholieren kunnen gebruiken om zinnen te markeren en deze vervolgens samen te vatten. Het samenvatten gebeurt echter op een manier waarbij de zinnen lexicaal en syntactisch identiek blijven. Helaas bieden deze softwarepakketten geen functie voor het vereenvoudigen van teksten. Volgens Tops e.a. (2018) is het belangrijk om deze software zo vroeg mogelijk in de schoolcarrière te introduceren, zodat scholieren er snel vertrouwd mee raken en het optimaal kunnen gebruiken in verdere studies. Hoewel Tops e.a. (2018) de handige aspecten van deze software benadrukt, is het te laat om deze software pas in het hoger onderwijs te introduceren.

Momenteel hebben scholieren met dyslexie in het middelbaar onderwijs beperkte tekstvereenvoudigingsfunctionaliteiten in de bestaande voorleessoftware. Dit onderstreept de noodzaak aan nieuwe erkende tools die tekstvereenvoudiging in het onderwijs mogelijk maken. Tools zoals Simplish en Reworkify kunnen een oplossing bieden. Hoewel Simplish oorspronkelijk Engelstalig is, is het inmiddels mogelijk om een vereenvoudigde tekst te genereren van Nederlandstalige teksten, al is deze functionaliteit betrekend voor niet-Engelstalige teksten. Reworkify is enkel in staat om Engelstalige teksten te vereenvoudigen. Er zijn echter maar weinig proof-of-concepts beschikbaar en de taalmodellen waarop deze applicaties werken zijn niet gekend om deze tools te kunnen repliceren. Bovendien zijn er geen API's beschikbaar om mee te werken.

Voor samenvatting zijn er echter meer tools beschikbaar. Enkele voorbeelden hiervan zijn Resoomer, Paraphraser, Editpad, Scribbr en Quillbot. Al zijn er onderzoe-

ken over geautomatiseerde lexicale en syntactische vereenvoudiging voor scholieren met dyslexie, het aantal onderzoeken over samenvatten voor deze doelgroep is schaars. Zoals eerder aangehaald is er wel onderzoek gedaan naar de verschillende manieren om een tekst samen te vatten, maar er is geen toepassing of onderzoek dat dit concreet uitwerkt. Stajner (2021) wijzen erop dat toepassingen voor tekstvereenvoudiging regelmatig als *showcase* van de technologie ontwikkeld worden en zelden tot weinig rekening houden met gepersonaliseerde samenvatting om zo rekening te houden met de verschillende noden.

Er zijn weinig toepassingen beschikbaar om wetenschappelijke artikelen te vereenvoudigen, maar er bestaan gratis en betalende toepassingen. Zo reiken SciSpace⁶ en Scholarcy ATS specifiek voor wetenschappelijke artikelen aan. en Scholarcy⁷. Hero vloeide verder uit het onderzoek van Bingel e.a. (2018), waarbij de onderzoekers een toepassing voor gepersonaliseerde ATS voor kinderen met dyslexie ontwikkelden. Hoewel Hero een oplossing aanreikt, slaagt de toepassing er niet in om wetenschappelijke artikelen te vereenvoudigen. Wel kunnen scholieren deze browserextensie gebruiken voor selecte Engelstalige nieuwssites. Tabel 2.9 geeft een overzicht van prevalent tools die momenteel tekstvereenvoudiging aanbieden.

⁶<https://typeset.io/>

⁷<https://www.scholarcy.com/>

Tool	Algemene functionaliteit
Sprintplus Kurzweil3000 Alinea Suite IntoWords TextAid	Voorleessoftware met ondersteuningsmogelijkheden voor moeilijke woordenschat
Resoomer Paraphraser Editpad Scribbr Quillbot	Samenvattingstool
SciSpace Scholarcy	Samenvattingstool specifiek voor wetenschappelijke artikelen.
Simplish Rewordify	Tool voor tekst vereenvoudiging met bijhorende analyse

Tabel 2.9: Overzicht van gekende voorleessoftware, tekstvereenvoudigings- en samenvattingstools die intuïtief zijn ontwikkeld voor de eindgebruiker (leerkracht of scholier).

Ontwikkelaars kunnen ook zelf aan de slag gaan. Zo beschikt Huggingface een breed scala aan API's en tools die gemakkelijk te downloaden en trainen zijn voor *pretrained* modellen voor veelvoorkomende NLP-taken, zoals *text classification*, taalmodellering en samenvatting. Tekstvereenvoudiging is in mindere mate aanwezig. Verder kunnen ontwikkelaars deze modellen *finetunen* op specifieke datasets om modellen te bouwen voor gepersonaliseerde NLP-taken. Ten slotte geeft tabel 2.10 taalmodellen van HuggingFace met hun respectievelijke casus.

Taalmodel	Specifieke casus
Google Pegasus	Samenvattingstaken voor kort tot middelgrote documenten.
Longformer Encoder-Decoder	Samenvatting van lange wetenschappelijke artikelen
Haining Scientific Abstract Simplification ⁸	Het lexicaal vereenvoudigen van wetenschappelijke artikelen.
BART Large Scientific Summarisation ⁹	Het samenvatten van wetenschappelijke artikelen.
T5 finetuned text simplification model ¹⁰	Tekstvereenvoudiging voor algemeen gebruik.
Keep It Simple ¹¹	Ongesuperviseerde tekstvereenvoudiging met ATS.

Tabel 2.10: HuggingFace beschikbare en ge-finetunedede taalmodellen.

Dankzij de sterke evolutie in data en AI, kon de grootte van deze taalmodellen sterk vergroten. Zo is GPT-3 een opkomend *Large Language Model* of LLM. OpenAI ontwikkelde dit taalmodel en paste daarvoor een tweestapsleerparadigma toe (C. Li, 2022; Radford e.a., 2019).

- Allereerst komt er een ongesuperviseerde training aan bod met een *language modeling goal*. Ontwikkelaars trainden dit model op niet-gecategoriseerde data van het internet met datasets zoals Common Crawl, WebText2, Books1, Books2 en Wikipedia.
- Tenslotte finetunen de ontwikkelaars dit verder. Om een correcte respons van het model te krijgen, passen de ontwikkelaars *reinforcement learning* toe.

GPT-3 beschikt over meerdere versies, waaronder GPT-3.5 die als engine dient voor ChatGPT. Omdat onbegrijpelijke en ontoegankelijke zinsstructuren niet alleen voor leken, maar ook voor vakexperten een obstakel vormen, is het belangrijk om te benadrukken dat GPT-3.5 gericht is op conversationele doeleinden, terwijl GPT-3 in het algemeen bedoeld is om met hoogstens één prompt te werken (Hubbard & Dunbar, 2017; McNutt, 2014).

Volgens Gooding (2022) kan een toepassing zoals ChatGPT de leesbaarheid van een tekst op twee manieren bevorderen. Ten eerste kan het unieke versies van een

tekstvereenvoudiging leveren, waardoor de gebruiker uit meerdere vereenvoudigingen kan kiezen. Figuur 2.6 illustreert dit verder. Zo geeft Gooding aan dat het taalmodel de invoertekst op meerdere manieren kan vereenvoudigen. Verder bespreekt Gooding in figuur 2.7 de prompt-gebaseerde evaluatiemaniere van GPT-3. Het model kan de verschillen tussen de gegenereerde teksten begrijpen en de gebruiker feedback geven. In figuur 2.5 laat Gooding zien hoe het model lexicale, conceptuele en syntactische vereenvoudiging kan toepassen op een paragraaf uit een boek van de Russische schrijver Dostoevsky.



Can you simplify the following paragraph:

At some ideas you stand perplexed, especially at the sight of sin. And you must ask yourself whether to combat it with force or with humble love. Always decide to combat it with humble love. If you decide this once and for all, you can conquer the whole world! Humble love is a frightening force, of which there is nothing stronger.



Here is a simpler version of the paragraph:

When you see something bad, like sin, you might be confused. Ask yourself if you should use force or love to fight it. Always choose love. If you do, you can win against anything. Love is the strongest power.

Figuur (2.5)

Afbeelding van Gooding 2022



Can you simplify the following paragraph in three different versions:

At some ideas you stand perplexed, especially at the sight of sin. And you must ask yourself whether to combat it with force or with humble love. Always decide to combat it with humble love. If you decide this once and for all, you can conquer the whole world! Humble love is a frightening force, of which there is nothing stronger.



Here are three simplified versions of the paragraph:

1. When you see something bad, like sin, you might be confused. Ask yourself if you should use force or love to fight it. Always choose love. If you do, you can win against anything. Love is the strongest power.
2. When you encounter ideas that confuse you, especially when it comes to sin, you may be unsure how to react. You must ask yourself if it is better to fight back with force or with humble love. It is always better to choose humble love. If you make this decision once and for all, you can overcome anything. Humble love is a powerful and intimidating force that is stronger than anything else.
3. If you see something bad, like sin, and you don't know what to do, remember that love is always the answer. Choose love over force. If you do, you will be able to conquer anything. Humble love is the most powerful force in the world.

Figuur (2.6)

Afbeelding van Gooding 2022



Which of the three simplified versions is the simplest?



Of the three simplified versions, the first one is the simplest. It uses the shortest and simplest words and phrases to convey the main idea of the original paragraph, which is that humble love is the strongest force and can be used to overcome anything. The other two versions add more details and examples, which make them slightly more complex than the first version. However, all three versions are simpler than the original paragraph and effectively convey the same idea.

Figuur (2.7)

Afbeelding van Gooding 2022.

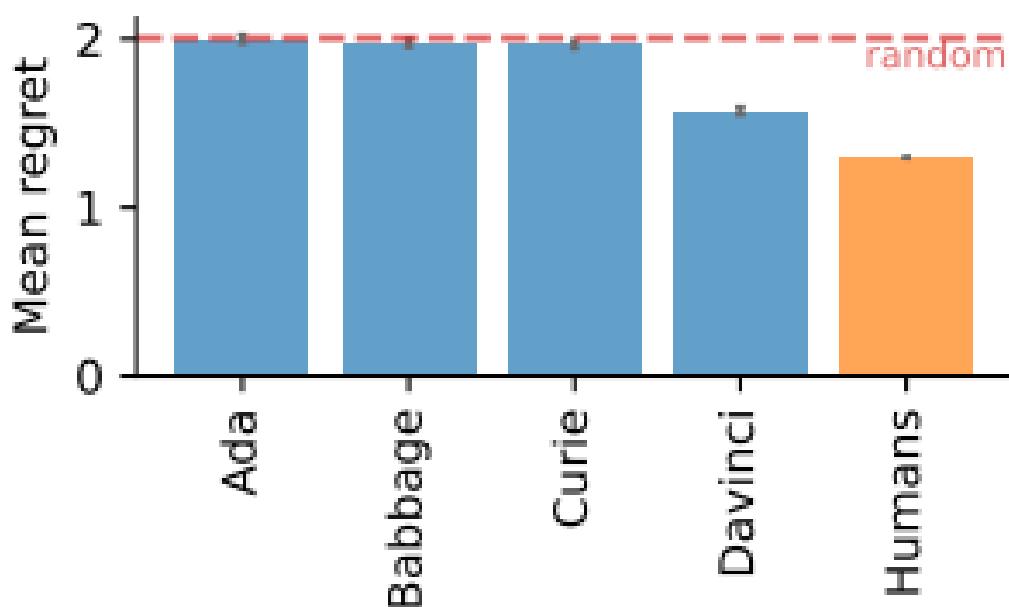
OpenAI reikt documentatie aan voor het GPT-3 taalmodel. Daarin vermelden ze vier *engines*, namelijk Davinci, Curie, Babbage en Ada. In maart 2023 kwam daar een vijfde engine bij, namelijk GPT-3 Turbo die fungeert als achterliggende engine voor Chat-GPT. Davinci-003 is het meest geavanceerde model, geschikt voor taken zoals het schrijven van essays en het genereren van code, en levert de meest menselijke antwoorden. Curie is goed in nuance, maar minder menselijk dan Davinci, terwijl Ada en Babbage minder krachtig zijn en beter geschikt zijn voor eenvoudige taken zoals het aanvullen van tekst en sentimentanalyse (Greg e.a., 2023). Deze engines gebruiken dezelfde set hyperparameters, die ontwikkelaars kunnen aanpassen. Tabel 2.11 somt deze parameters verder op.

Parameter	Omschrijving	Mogelijke waarden
model	De GPT-3 engine die ontwikkelaars kunnen gebruiken.	davinci, curie, babbage, ada, text-davinci-002, text-curie-001, text-babbage-001, text-ada-001, davinci-codex
temperature	De gulzigheid van het generatief model. Een lagere waarde kan voor-spelbare tekst teruggeven. Hogere waarden daarentegen kunnen onverwachtse tekst teruggeven, wat beter werkt bij creatieve toepassingen.	Een kommagetal tussen 0 en 1.
max tokens	Het maximaal aantal tokens (woorden of subwoorden) dat het generatief model kan teruggeven.	Een getal tussen 1 and 2048.
top-p	Vergelijkbaar met temperature, maar deze waarde onderhoudt de <i>probability distribution</i> voor <i>common tokens</i> . Hoe lager de waarde, hoe hoger de woordfrequentie van de gegenereerde tekst. Toepassingen gericht op nauwkeurigheid maken beter gebruik van hoge waarden.	Een kommagetal tussen 0 en 1.
stop	Een tekstwaarde (woord/symbool) tot waar het model zal genereren.	String-waarden
presence penalty	Factor die bepaalt hoe regelmatig woorden voorkomen	Kommagetallen tussen 0 en 1

Tabel 2.11: Tabel met alle GPT-3 parameters.

Hoewel onderzoeken rond GPT-3 nog volop in ontwikkeling zijn, bestaan er vergelijkende onderzoeken naar de mogelijkheden van dit LLM. Onderzoek van Binz en Schulz (2023) wijst uit dat de mean-regret score kan dienen als maatstaf om de menselijkheid van antwoorden te beoordelen. Deze studie wees verder uit dat deze modellen capabel zijn om menselijke antwoorden te produceren, zoals geïllustreerd in figuur 2.8. Uit het experiment van Tanya Goyal (2022) blijkt dat zero-shot samenvattingen met GPT-3 beter presteren dan gefinetunedede modellen. (C.

Li, 2022) benadrukt dat GPT-3 overkill is voor taken zoals sentimentanalyse -of classificatie. Daarvoor halen de onderzoekers aan om een kleinschaliger taalmodel te gebruiken. Deze taalmodellen beschikken ook over een grotere ecologische voetafdruk, waarvoor onderzoeken van Simon (2021) en Strubell e.a. (2019) deze praktijk ook afraden. Ten slotte bestaan er al enkele tools die gebruikmaken van de GPT-3 API, waaronder Jasper AI en ChatSonic zoals aangehaald in het onderzoek van Mottesi (2023). Experten zoals Garg (2022) en Roose (2023) halen het GPT-3 model en ChatGPT aan als de toekomst voor gepersonaliseerde en adaptieve uitleg aan scholieren. Bing Chat biedt een extra dat revolutionair kan zijn bij het opzoeken van uitleg voor zoektermen, zonder het verlies aan bronvermelding. Dankzij de personalisering van de prompts biedt GPT-3 mogelijkheden aan voor toepassingen in het onderwijs, aldus Garg (2022) en Roose (2023).

B**Figuur (2.8)**

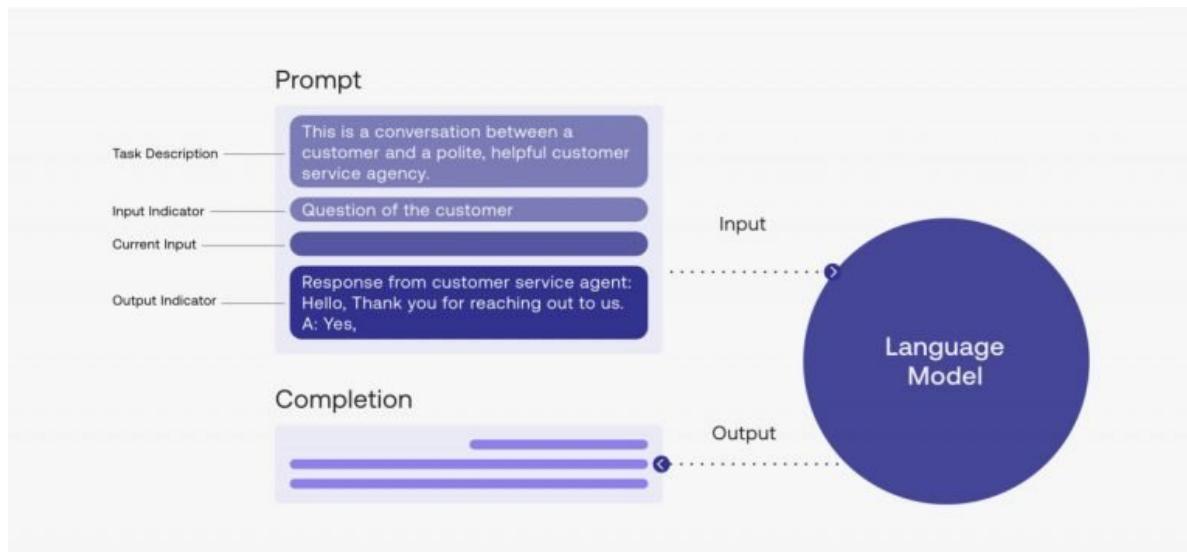
Een experiment op de *mean-regret* van GPT-3 engines uit Binz en Schulz (2023).

Met neurale netwerken kan het taalmodel patronen in de input herkennen. Deze patronen dienen om voorspellingen te maken over de output (Liu e.a., 2020). Via prompts hebben mensen toegang tot krachtige taalmodellen, zoals GPT-3 of BERT (Harwell, 2023; McFarland, 2023). Figuur 2.9 illustreert de werking van deze vaardigheid. Onderzoek van Liu e.a. (2020) benadrukt het belang van goed opgestelde prompts. Hiervoor kunnen eindgebruikers de technieken in tabel 2.12. Zo moeten

Prompttechniek	Bron
Duidelijke scope	(McFarland, 2023)
Specifieke sleutelwoorden	(McFarland, 2023)
De context waarin de vraag zich afspeelt.	(McFarland, 2023)
Gepersonaliseerde keuzes	(McFarland, 2023)

Tabel 2.12: Technieken voor concrete en goed opgestelde prompts.

deze werk kunnen produceren op maat van het doel. Zo benadrukt de onderzoeker dat een prompt concreet moet zijn. Bij het opstellen van een prompt voor een zoekopdracht is het cruciaal om voldoende parameters op te nemen om te voorkomen dat het model te algemeen blijft en afwijkt van de intentie van de gebruiker. Effectieve prompt engineering voor AI leidt tot hoogwaardige trainingsgegevens, waardoor het AI-model nauwkeurige voorspellingen en beslissingen kan maken (Liu e.a., 2020).



Figuur (2.9)

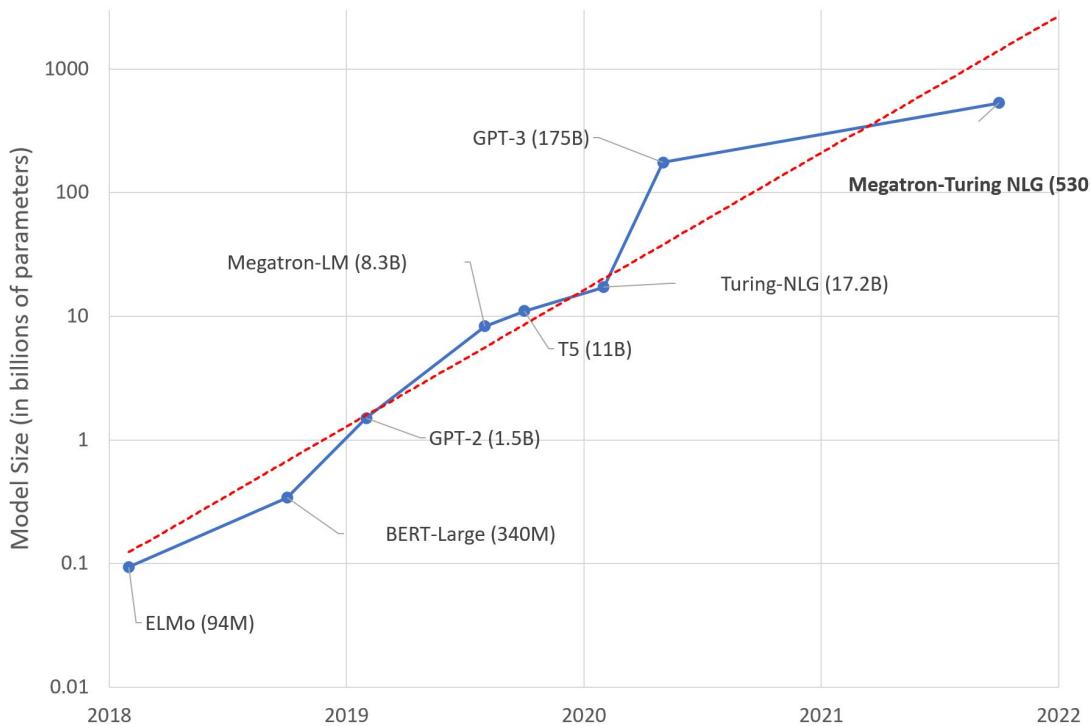
De werking van *prompt engineering* volgens McFarland (2023)

Om prompts te kunnen hergebruiken, bouwden ontwikkelaars *prompt patterns* op. Deze patronen bieden herbruikbare oplossingen voor veelvoorkomende problemen in een bepaalde context, vooral bij de interactie met LLM's. White e.a. (2023) benoemt vier *prompt patterns*:

- *Intent prompts* waarbij een LLM een instructie krijgt met een specifiek verwacht antwoord.

- *Restriction prompts* die het antwoord van een LLM inperkt. Deze pattern is noodzakelijk om een LLM binnen de lijnen te houden.
- *Contextualization prompts* verzekeren dat de output van een LLM relevant is. De prompt bevat de context.
- *Expansion/reduction prompts* genereren een beknopte output met voldoende details.

BERT is een meertalige LLM die gebruikmaakt van *contextual word embeddings*. Onderzoekers trainden het model op 110 miljoen parameters uit 104 verschillende talen, waaronder Nederlands. Voor de Nederlandse taal zijn er twee varianten van BERT beschikbaar: RobBERT en BERTje. GPT-3 en BERT beschikken over een verschillende architectuur. Zo is volgens Mottesi (2023) GPT-3 een autoregressief model dat alleen rekening houdt met de linkercontext bij het voorspellen of genereren van tekst. BERT daarentegen is een bidirectioneel model, waarbij het model zowel de linker- als de rechtercontext in overweging neemt. De bidirectionele werking van BERT is geschikt voor sentimentanalyse, waarbij begrip van de volledige zin-context noodzakelijk is. Omdat GPT-3 toegang heeft tot meer informatie (45TB) dan BERT (3TB), kan dit een voordeel bieden tijdens taalbewerkingen zoals vereenvoudigen, parafraseren of vertalen. Ten slotte verschillen de LLM's ook qua grootte. Hoewel beide modellen erg groot zijn, is GPT-3 aanzienlijk groter dan BERT, mede door de uitgebreide trainingsdatasetgrootte (Brown e.a., 2020). Er is echter een nieuw generatief taalmodel genaamd LLaMa, dat sterker is dan GPT-3 en vergelijkbare modellen, terwijl het slechts tien keer minder parameters gebruikt. Helaas is LLaMa momenteel nog niet beschikbaar als online webtoepassing of API (Hern, 2023; Touvron e.a., 2023). Figuur 2.10 toont de evolutie van *pre-trained* taalmodellen. Zo volgt de LLM-performantie ten opzichte van het aantal parameters van een LLM een lineaire functie.

**Figuur (2.10)**

De evolutie van LLM's (Simon, 2021)

Microsoft en OpenAI werken nauw samen. Zo gebruikt Bing Chat ook het GPT-3 taalmodel. Met de Prometheus-technologie kan deze chatbot verder bouwen en verwijzingen bieden naar andere websites (Ribas, 2023). Zo maakt Bing Chat verwijzingen naar bestaande online documenten dankzij de Bing index-, ranking- en antwoordresultaten. Prometheus combineert dit met het redeneervermogen van OpenAI's GPT-modellen. Via de *Bing Orchestrator* genereert Prometheus iteratief een set *internal queries* om zo binnen de gegeven gesprekscontext nauwkeurige antwoorden op gebruikersvragen te bieden (Ribas, 2023). Bing Chat is beschikbaar als webpagina en browserextensie voor Microsoft Edge. Het gebruik van extraherende en abstraherende samenvattingen maakt de chatbot interessant, al bestaat er nog te weinig onderzoek naar de credibiliteit en correctheid van de verstrekte verwijzingen. Daarnaast beschikt Bing Chat niet over een API, waardoor ontwikkelaars geen commandlinetoepassingen met deze technologie kunnen maken.

2.6. De valkuilen bij AI en NLP.

AI en ML zijn volop in groei. NLP gebruikt AI en ML om menselijke taal te verwerken, terwijl NLU deze technologieën gebruikt om menselijke taal te begrijpen. Hoewel deze technologieën veelbelovend zijn, moeten AI-ontwikkelaars rekening houden met veelvoorkomende en genegeerde uitdagingen en valkuilen (Khurana e.a.,

2022; Roldós, 2020; Sciforce, 2020). Deze sectie beantwoordt de volgende onderzoeksfrage:

- Met welke valkuilen bij taalverwerking met AI moeten ontwikkelaars rekening houden?

Het ontwikkelen van NLP- en NLU-toepassingen is duur en vormt een obstakel voor veel IT-professionals vanwege factoren zoals het gebrek aan NLP-expertise, de kwaliteit en kwantiteit van data, de integratie en deployment van modellen en de transparantie van modellen (IBM, 2022). Bij de ontwikkeling en finetuning van een NLP-toepassing met AI verkiezen software-ontwikkelaars *black-box* modellen. Hoewel het verschil in nauwkeurigheid minimaal is, wordt de afweging gemaakt op basis van de transparantie van het model. Na een transformatie wordt niet aangegeven waarom specifieke transformaties zijn uitgevoerd, zoals het vervangen van een woord door een eenvoudiger synoniem. *White-box* taalmodellen zijn schaars (Sikka & Mago, 2020).

Sequence labeling kan worden bemoeilijkt door homoniemen, zoals beschreven in onderzoek door Roldós (2020). Een voorbeeld hiervan is het woord 'bank', dat voor de machine niet duidelijk aangeeft of het gaat om de financiële instelling of het meubelstuk. Methoden zoals Word Sense Disambiguation (WSD), PoS-tagging en contextual embeddings kunnen de betekenis van woorden bepalen op basis van de context (Eisenstein, 2019; Liu e.a., 2020). Het verbeteren van NLP-systeem met synoniemen en antoniemen kan worden bereikt door het gebruik van candidate generation en synonym detection, terwijl meertalige transformers zoals BERT een oplossing bieden voor de beperkte toepassingen in andere talen dan het Engels (Dandekar, 2016; Roldós, 2020).

Verschillende studies hebben aangetoond dat tekstvereenvoudiging bedoeld is om gelijke kansen te bieden aan iedereen. Bij het ontwikkelen van adaptieve tekstvereenvoudigingstoepassingen is het echter belangrijk om ethische overwegingen en bewustzijn van de behoeften van de eindgebruiker in acht te nemen, zoals beschreven in onderzoeken van Niemeijer e.a. (2010), Xu e.a. (2015), en Gooding (2022). Om de eindgebruiker meer controle te geven, moet deze de keuze hebben om te kiezen welke delen van de tekst vereenvoudigd moeten worden, wat kan worden bereikt door synoniemen te gebruiken of zinnen te markeren die moeilijk te begrijpen zijn.

Hoewel iedereen kan converseren met een chatbot, vereist het verkrijgen van gepaste en verwachte antwoorden een doordachte input. Onnauwkeurige prompts of een gebrek aan trainingsdata kunnen leiden tot onjuiste output. Door het gebruik van conditionele expressies of het finetunen van hyperparameters kan echter

de betrouwbaarheid van de antwoorden worden vergroot (Jiang, 2023; Miszczak, 2023).

Het beoordelen van vereenvoudigde teksten vereist de nodige opvolging van ontwikkelaars in vergelijking met andere ML- of NLP-taken. Evaluatiemetrieken zoals ROUGE en BLEU zijn beperkt omdat ze geen rekening houden met de semantiek tussen een referentietekst en een vereenvoudigde of samengevatte tekst. Om dit probleem op te lossen, beveelt Fabbri e.a. (2020) aan dat ontwikkelaars menselijke evaluatie inschakelen om de vereenvoudigde tekst van een taalmodel te beoordelen. De onderzoekers dringen aan op verdere studie naar nieuwe standaarden en beste praktijken voor betrouwbare menselijke beoordeling. Bovendien moeten de doelgroepen voor wie de tekst wordt vereenvoudigd, nauw betrokken worden bij het proces (Iskender e.a., 2021). Tabel 2.13 somt de aangehaalde struikelblokken bij de ontwikkeling van NLP-toepassingen op.

Probleem	Oplossing
Dure ontwikkeling en onderhoud van taalmodellen	Voorkeur voor black-box modellen bij ontwikkeling en finetuning. API's kunnen als alternatief dienen op zelf-gehoste taalmodellen.
Homoniemen kunnen sequence labeling bemoeilijken	Word Sense Disambiguation, PoS-tagging en contextual embeddings.
Paternalisme	Ontwikkeling van gepersonaliseerde tekstvereenvoudigingstoepassingen moeten de eindgebruiker meer controle geven, zoals het kiezen welke delen van de tekst vereenvoudigd moeten worden, het gebruik van synoniemen of het markeren van zinnen die moeilijk te begrijpen zijn.
Onnauwkeurige prompts	Gebruik van conditionele expressies bij prompts of one-shot summary uitvoeren.
Onnauwkeurige evaluatie van tekstvereenvoudiging	Menselijke evaluatie toepassen of gebruik maken van ROUGE-L metrieken die wel de semantiek in acht nemen.

Tabel 2.13: Samenvattend schema met vaak voorkomende struikelblokken bij NLP-toepassingen.

2.7. Conclusie

Begrijpend lezen is voor scholieren met dyslexie in de derde graad van het middelbaar niet enkel moeizaam, maar gaat verder dan dat. Deze scholieren kunnen moeite ondervinden bij het decoderen en automatiseren van woordherkenning. MTS en aangepaste opmaakopties bieden bewezen voordelen voor scholieren met dyslexie. Zo kunnen de MTS-technieken, besproken in 2.6 dienen als afroetscriteria. Daarnaast neemt de leesbaarheid van wetenschappelijke artikelen neemt af. Het gebruik van vakjargon, ingewikkelde woordenschat en moeizame syntax sluiten een algemeen publiek uit en maken het enkel mogelijk voor wetenschappelijk geletterden om deze artikelen te lezen.

Een uniform formaat en de evolutie van LLM's bieden kansen voor gepersonaliseerde ATS aan, waarbij experts tactieken hebben ontwikkeld om teksten te vereenvoudigen op maat voor scholieren met dyslexie. Gerichte menselijke vergelijkingen en leesbaarheidsformules dienen als evaluatie voor de vereenvoudigde teksten door ATS.

Er zijn taalmodellen beschikbaar in de vorm van API's of open-source software die deze transformaties kunnen uitvoeren. De overheid leent voornamelijk leessoftware uit, maar LLM's bieden mogelijkheden aan voor gepersonaliseerde ATS. Ontwikkelaars moeten zich echter bewust zijn van het feit dat andere taalmodellen zoals BERT voor taken zoals semantische analyse soms betere resultaten leveren met minder rekenkracht.

3

Methodologie

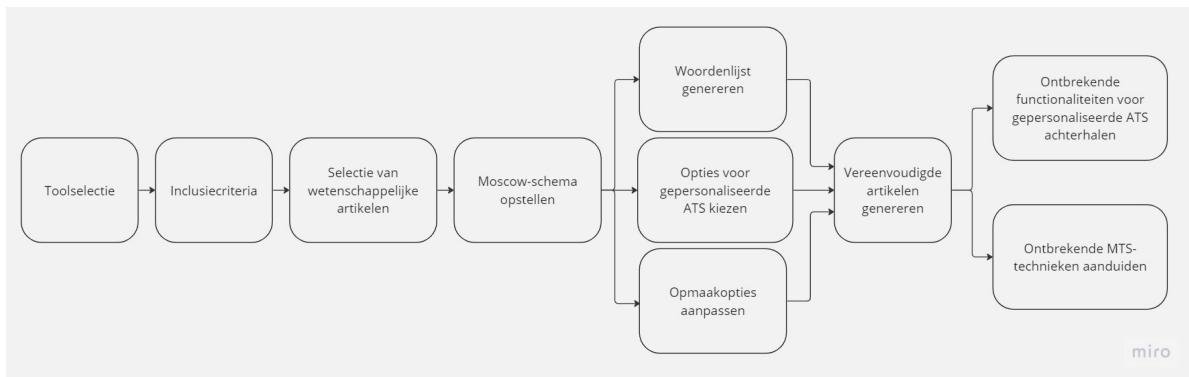
Om een antwoord te vormen op de onderzoeksvraag, moet de vergaarde kennis uit de literatuurstudie in drie onderzoeksmethoden. Eerst staat het onderzoek stil bij de vereiste functionaliteiten om gepersonaliseerde ATS te kunnen realiseren. Vervolgens achterhaalt het onderzoek het geschikte taalmodel voor gepersonaliseerde ATS. Tenslotte volgt de ontwikkeling van een prototype voor ATS-verenvoudiging van wetenschappelijke artikelen. Zo doelt dit onderzoek om de haalbaarheid voor een toepassing voor gepersonaliseerde ATS te achterhalen aan de hand van een prototype. Dit prototype moet in staat zijn om wetenschappelijke artikelen te vereenvoudigen op maat voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

3.1. Requirementsanalyse

Om het ontwikkelingsproces van het prototype gericht te sturen, moet het onderzoek MTS- en ATS-technologieën in bestaande tools nagaan. Zo gebeurt het verkennen en experimenteren op ATS-technieken bij beschikbare tools door een kwalitatief onderzoek in de vorm van een requirementsanalyse. Het resultaat van deze onderzoeksfase is een Moscow-schema dat de benodigde functionaliteiten voor een toepassing met ATS definieert, met als doel een vergelijkbare toepassing aan te bieden voor gepersonaliseerde ATS van wetenschappelijke artikelen met de kwaliteiten van gepersonaliseerde MTS. Daarnaast achterhaalt de onderzoeksfase de ontbrekende MTS-functionaliteiten die tabel 2.6 in de literatuurstudie uitwees. De geteste toepassingen, opgesomd in tabel 3.1, beschikken over (gepersonaliseerde) ATS-technieken. Deze lijst omvat erkende toepassingen van de overheid en toepassingen die leerkrachten of scholieren kunnen gebruiken om teksten te vereenvoudigen. Met deze onderzoeks methode kan het onderzoek een antwoord geven op de volgende twee deelvragen van het onderzoek.

- Welke functies ontbreken AI-toepassingen om geautomatiseerde tekstvereenvoudiging mogelijk te maken voor scholieren met dyslexie in de derde graad middelbaar onderwijs?
- Welke manuele methoden voor tekstvereenvoudiging komen niet in deze tools voor?

Figuur 3.1 toont de flowchart om de requirementsanalyse te kunnen uitwerken.



Figuur (3.1)

Het benodigde stappenplan bij de requirementanalyse.

Allereerst start het onderzoek met een toolselectie. Zoals aangewezen in sectie 2.5, leent de overheid vijf softwarepakketten uit aan middelbare scholen. Echter neemt de requirementsanalyse drie van deze vijf in de analyse op, want hun functionaliteiten zijn passend voor deze doelgroep. De overige twee zijn minder aanwezig in het onderwijs. Daarnaast toont een zoekopdracht aan dat deze tools geen lexicale vereenvoudiging toepassen.

Naast deze erkende softwarepakketten kunnen online beschikbare tools ook scholieren met dyslexie ondersteunen bij het begrijpend lezen van wetenschappelijke artikelen met ATS, zoals bewezen in Bingel e.a. (2018). Daarom betreft de requirementsanalyse enkel tools met onderschreven ATS-functionaliteiten en laat daarmee pure samenvattingstools erbuiten. Tabel 3.1 toont een overzicht van de te experimenteren tools.

Erkende software	Online beschikbare tools
Sprintplus (E1)	Simplis (O1)
Kurzweil3000 (E2)	SciSpace (O2)
AlineaSuite (E3)	Rewordify (O3) ChatGPT (O4) Bing Chat (O5)

Tabel 3.1: Shortlist van uit te testen tools en toepassingen voor tekstvereenvoudiging.

Vervolgens bouwt het onderzoek een lijst op van de toetsingscriteria. Zo dienen de MTS-technieken uit tabel 2.3 en tabel 2.5 als bouwstenen voor het opstellen van de toetsingscriteria. Tabel 3.2 geeft een opsomming van de MTS-technieken waaraan tools moeten voldoen. Daarnaast dient dit schema om een inschatting van de functionaliteiten van deze tools te maken.

MTS-techniek	Functionaliteit
Lexicale vereenvoudiging	<ul style="list-style-type: none"> Gepersonaliseerde LS, ofwel woordenschat dat niet te hoog gegrepen is. Gekende woordenschat mag blijven. Woorden met minder lettergrepen gebruiken Extra uitleg schrijven bij zinnen Paragrafen herschrijven zodat ze eerst uitleg geven op een high-level niveau, vervolgens lagen van complexiteit toevoegen om de lezer te begeleiden Woordenlijst aanmaken Idiomen vervangen door eenvoudigere synoniemen
Syntactische vereenvoudiging	<ul style="list-style-type: none"> Zinnen inkorten Verwijswoorden aanpassen Voorzetseluitdrukkingen aanpassen Samengestelde werkwoorden aanpassen Actieve stem toepassen Enkel regelmatige werkwoorden gebruiken
Structurele aanpassingen	<ul style="list-style-type: none"> Achtergrondkleur aanpassen Woord- en karakterspatiëring Consistente lay-out Duidelijk zichtbare koppenstructuur Huidige positie benadrukken Waarschuwingen geven omtrent formulieren en sessies Inhoud visueel groeperen Tekst herschrijven als tabel Tekst herschrijven als opsomming

Tabel 3.2: Richtlijnen waarop het onderzoek de toepassingen afroeft in de requirementsanalyse.

Als realistisch testmateriaal, maken de experimenten gebruik van twee gepubliceerde wetenschappelijke artikelen. Zo kunnen deze artikelen relevant zijn voor leerkrachten om aan scholieren in de derde graad van het middelbaar onderwijs te geven als leesvoer. Beide artikelen volgen de kenmerken van een wetenschappelijk artikel, zoals beschreven in tabel 2.3. Daarnaast gebruiken ze vakjargon en wetenschappelijke concepten in een compact formaat. Tabel 3.3 geeft een overzicht van de twee artikelen en een bijhorende bronvermelding.

Titel	Bronvermelding
De controle op het gebruik van algoritmische surveillance- onder druk? Een exploratie door de lens van de relationele ethiek	(Van Brakel, 2022)
Nederland versus België: verschillen in economische dynamiek en beleid.	(Sleuwaegen, 2022)

Tabel 3.3: Bronvermeldingen voor de twee wetenschappelijke artikelen.

Om een overzicht te hebben van de functionaliteiten volgens prioriteit, bouwt het onderzoek een Moscow-schema vanuit de opgestelde richtlijnen. Zo komen belangrijke functionaliteiten, die nodig zijn om gepersonaliseerde tekst vereenvoudiging met ATS mogelijk te maken, in de categorie *must-haves* terecht. Alle vereiste functionaliteiten om (gepersonaliseerde) ATS mogelijk te maken, moet als *must-have* in het Moscow-schema voorkomen. Irrelevante functionaliteiten binnen de scope van een prototype of niet toepasselijk voor de doelgroep plaatst het onderzoek als *wont-have*.

MoSCoW-principe	Functionaliteit
Must-have	<p>Gepersonaliseerde LS, ofwel woordenschat dat niet te hooggegrepen is. Gekende woordenschat mag blijven.</p> <p>Woorden met minder lettergrepen gebruiken.</p> <p>Woordenlijst aanmaken na handmatige CWI.</p> <p>Wetenschappelijke artikelen in PDF-vorm opladen.</p> <p>Structurele aanpassingen toepassen op de oorspronkelijke tekst.</p> <p>Personaliseerbare opmaakopties, waaronder lettertype - en grootte aanpassen, tekstformaat aanpassen, achtergrondkleur aanpassen.</p> <p>Duidelijk zichtbare koppenstructuur.</p> <p>Tekst herschrijven als opsomming.</p>
Should-have	<p>Tekst-analyse</p> <p>Extra (in-line) uitleg schrijven bij moeilijke woordenschat.</p> <p>Personaliseerbare PDF- of Word-document lay-out.</p> <p>Uitvoer als PDF of Word-bestand teruggeven.</p> <p>Wetenschappelijke artikelen in PDF-vorm opladen met OCR.</p> <p>Tekstanalyse voor en na de vereenvoudiging aanbieden.</p>
Could-have	<p>Gebruikersfeedback</p> <p>Woordenschat genereren na automatische CWI.</p> <p>Huidige positie benadrukken.</p> <p>Waarschuwingen geven omtrent formulieren en sessies.</p> <p>Enkel regelmatige werkwoorden gebruiken.</p> <p>Extraherende samenvatting</p> <p>Abstraherende samenvatting</p> <p>Tekst herschrijven in tabelvorm</p>
Wont-have	<p>Mobiele versie of <i>responsive design</i>.</p> <p>Audio-uitvoer</p> <p>Integratie met externe toepassingen, waaronder spellcheckers.</p>

Tabel 3.4: Het Moscow-schema voor de requirementsanalyse.

Vervolgens komen experimenten rond de opties rond gepersonaliseerde ATS aan bod. Eindgebruikers moeten wetenschappelijke artikelen kunnen opladen. Eindgebruikers kunnen enkel pdf's inladen bij SprintPlus, Kurzweil3000, AlineaSuite, Simplish, Rewordify en SciSpace. In tegenstelling tot Bing Chat en ChatGPT waarbij *plain-text* of een link naar het wetenschappelijk artikel de enige vormen van invoer zijn. De tekstinhoud uit de PDF extraheren gebeurt dan manueel en deze tekst dient daarna als invoer voor de chatbot. Beide chatbots krijgen de prompt, gevolgd door een stuk van het wetenschappelijk artikel. Met zes verschillende prompts kan het onderzoek de functionaliteit om LS, SS of structurele aanpassingen op een tekst door een toepassing achterhalen. Tabel 3.5 vermeldt de toegepaste prompts. Toepassingen krijgen eerst een link van het wetenschappelijk artikel, zoals aangegeven in figuur C.2. Als de toepassing hier niet over beschikt, dan krijgt de chatbot de tekstinhoud van het wetenschappelijk artikel in *plain-text* mee.

Naam	Prompt
P1	Vereenvoudig deze tekst.
P2	Vereenvoudig deze tekst voor studenten (16-18 jaar) door moeilijke woorden te vervangen, vakjargon te schrappen, woorden langer dan 18 letters te vervangen, acroniemen voluit te schrijven, een woord slechts eenmaal door een synoniem te vervangen, korte uitleg te geven wanneer dat nodig is, en percentages te vervangen.
P3	Vereenvoudig een tekst door deze op te delen in kortere zinnen van maximaal tien woorden. Verander voornaamwoorden als 'zij', 'hun' of 'hij' in namen. Vervang complexe zinsconstructies en voorzetzelzinnen door eenvoudiger alternatieven, maar laat ze ongewijzigd als er geen eenvoudiger optie beschikbaar is.
P4	Schrijf de tekst als opsomming.
P5	Schrijf de tekst in tabelformaat.
P6	Genereer op basis van deze tekst een woorden- en synoniemenlijst.

Tabel 3.5: De toegepaste GPT-3-prompts in de requirementsanalyse.

Daarna voert het onderzoek experimenten rond gepersonaliseerde opmaakopties uit. Kurzweil, SprintPlus en AlineaSuite bieden opmaakopties aan in het instellingscherm. Zo kunnen eindgebruikers het lettertype, -kleur -en grootte en de ach-

tergrondkleur aanpassen naar keuze. Als een toepassing niet over opmaakopties beschikt, dan stopt het experiment rond opmaakopties voor die toepassing.

Ten slotte test het onderzoek de capaciteiten om het formaat van teksten aan te passen aan de hand van structurele aanpassingen. Zo vragen P4 en P5 specifiek naar een structurele aanpassing, terwijl P1, P2 en P3 minstens een doorlopende tekst als resultaat verwachten. Andere beschikbare tools missen checkboxen of keuzelijsten om deze keuze aan te reiken, waardoor het testen van deze functionaliteit niet mogelijk is.

3.2. Vergelijkende studie

Om wetenschappelijke artikelen met gepersonaliseerde ATS te vereenvoudigen, moet dergelijk toepassing gebruikmaken van een geschikt taalmodel. Zo moet de uitvoer van het prototype een vereenvoudigde versie van een wetenschappelijk artikel kunnen geven, specifiek voor de noden van scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Om de uitvoer van het prototype nauwkeurig af te stemmen, vereist het onderzoek een antwoord op de volgende vraag.

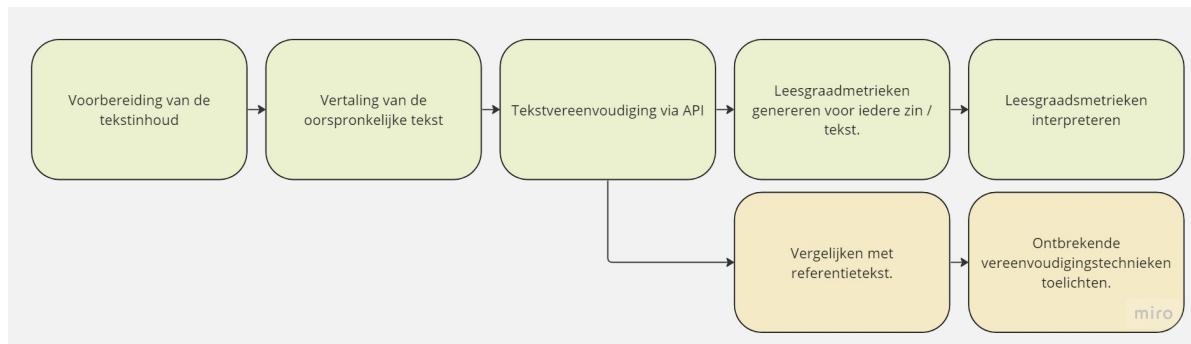
- Welk taalmodel is geschikt voor tekstvereenvoudiging met ATS van wetenschappelijke artikelen voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs, met dezelfde of gelijkaardige kwaliteiten als gepersonaliseerde tekstvereenvoudiging met MTS?

Er zijn weinig gespecialiseerde taalmodellen beschikbaar om wetenschappelijke artikelen te vereenvoudigen. Daarom beoordeelt de vergelijkende studie alle vermelde taalmodellen opgesomd in tabel 3.6.

Verwijzing	Taalmodel
T1	Haining Scientific Abstract Simplification
T2	BART-based Scientific Lay Summarizer
T3	Keep It Simple
T4	GPT-3

Tabel 3.6: Gebruikte taalmodellen in de vergelijkende studie

De vergelijkende studie bestaat uit vijf fasen, weergegeven op de *flowchart* in figuur 3.2. Zo vergelijkt deze onderzoeksfase de leesgraadscores van de oorspronkelijke wetenschappelijke artikelen zoals in sectie 3.1, met referentieteksten vereenvoudigd met MTS en teksten vereenvoudigd met ATS. Met een *mixed-methods*



Figuur (3.2)

Het gevolgde stappenplan voor de vergelijkende studie.

onderzoek kan de vergelijkende studie taalmodellen beoordelen op objectief en subjectief niveau.

De gebruikte wetenschappelijke artikelen zijn identiek aan de taalmodellen in tabel 3.3. Om realistisch referentiemateriaal te verkrijgen, schrijven twee leerkrachten en twee leerlingen zonder dyslexie zelf een vereenvoudiging van de twee wetenschappelijke artikelen met MTS. Deze vier personen baseren zich op vooraf meegekregen richtlijnen, toegelicht in de bijlage B.

Eerst volgt er een voorbereiding van de tekstinhoud. Allereerst haalt het script de inhoud van de map met wetenschappelijke artikelen op om deze vervolgens in een tekstbestand te plaatsen, zoals weergegeven in codeblok 3.1.

```
1 def add_newline_after_dot(input_file, output_file):
2     with open(input_file, 'r', encoding='utf-8') as file:
3         text = file.read()
4         text = re.sub(r'\d', '', text)
5         modified_text = text.replace('. ', '.\n')
6         with open(output_file, 'w', encoding='utf-8') as file:
7             file.write(modified_text)
8
9 folder_path = 'scripts\\pdf'
10 original_scientific_papers = [f for f in os.listdir(folder_path)]
11
12 for paper in original_scientific_papers:
13     input_file = folder_path + '/' + paper
14     output_file = folder_path + '/' + 'RE_' + paper
15     add_newline_after_dot(input_file, output_file)
```

Listing 3.1: Script voor fase 1 van de vergelijkende studie.

Vervolgens vertaalt het script de zinnen naar Engels. Het script, verwezen in script 3.2, doorloopt alle tekstbestanden en vertaalt de tekstinhoud met de `deep_translator` python-bibliotheek. Zo bekomt het script een csv-bestand met twee kolommen: alle Nederlandstalige en alle vertaalde Engelstalige zinnen van één wetenschappelijk artikel. Als separator gebruikt het csv-bestand een *pipe*-symbool.

```

1  output_csv = 'results.csv'
2
3  def translate_dutch_to_english(dutch_text_file):
4      with open(dutch_text_file, 'r', encoding='utf-8') as file:
5          dutch_sentences = file.readlines()
6          dutch_sentences = [sentence.strip() for sentence in dutch_sentences]
7
8      english_sentences = []
9      for sentence in dutch_sentences:
10         translated = GoogleTranslator(source='nl', target='en').translate(sentence)
11         english_sentences.append(translated)
12     df = pd.DataFrame({'Dutch': dutch_sentences, 'English': english_sentences})
13     df.to_csv(str(dutch_text_file).split('.')[0] + '.csv', index=False)
14
15
16 folder_path = 'scripts/pdf/'
17 original_scientific_papers = [f for f in os.listdir(folder_path)]
18
19 for paper in original_scientific_papers:
20     if paper.startswith('RE_') and paper.endswith('.txt'):
21         print(f'STARTING {paper}')
22         dutch_text_file = folder_path + paper
23         translate_dutch_to_english(dutch_text_file)
24

```

Listing 3.2: Script voor de tweede fase van de vergelijkende studie.

Na de vertaling van de tekstinhoud, stuurt het script API-calls voor iedere zin naar ieder taalmodel. Dit procesde taalmodellen aan om de inhoud van de wetenschappelijke artikelen te vereenvoudigen, weergegeven in listing 3.3. Allereerst vindt een tokenisatiefase plaats. Daarvoor gebruikt het script de *sentence tokenisation* van Spacy en de verwante *embedding models* weergegeven in tabel 3.7. Na de tokenisering voert het script API-calls uit. API-calls in de *scientific_simplify*-functie naar HuggingFace of OpenAI sturen de tekst naar de taalmodellen die de tekst verwerkt. De request naar de HuggingFace API bestaat uit de parameters weergegeven in tabel 3.8. Alle HuggingFace taalmodellen vereisen een laadfase. Daarom bevat de *API-call* een extra parameter, namelijk *wait_for_model*. Verder past dit script geen extra parameters van de taalmodellen aan.

Zoals aangehaald door Gooding (2022) kunnen prompt-gebaseerde testen verschillende resultaten krijgen, afhankelijk van de gegeven input. Daarom gebruikt het script drie verschillende prompts, gebaseerd op de MTS-technieken beschreven in tabel 2.5. Tabel 3.9 visualiseert de gebruikte prompts voor de testen met het GPT-3 model.

Zowel T1 als T4 gebruiken een *nul-temperature* en een *top-p* waarde van 90% om

vertrouwde antwoorden te krijgen. Daarnaast dient de top-p om een hoge woordfrequentie te verkrijgen, zoals aangegeven in 2.11. Bij T1 zijn deze twee parameters ingebakken in de functie. Nadien verwerken de taalmodellen iedere zin uit de tekst.

Ten slotte beantwoordt de HuggingFace of GPT API in JSON-formaat, bevattende de vereenvoudigde versie van de opgegeven zin. T1, T2 en T3 vereenvoudigen de Engelstalige zinnen, in tegenstelling tot T4 en verwante prompts die de Nederlands-talige zinnen vereenvoudigen. Om een *request failure* door een te lange input te voorkomen, breekt het script de volledige input op per 1000 tokens.

Taal	Embeddingsmodel
Nederlands	NL Core News Medium ¹
Engels	EN Core Web Medium ²

Tabel 3.7: Gebruikte SpaCy word-embeddings

Naam parameter	Waarde
Inputs	De oorspronkelijke zin. Enkel bij T1 komt 'simplify:' voor deze zin.
Max length	De lengte van de oorspronkelijke zin + 10 tokens.
Wait for model	Altijd ingesteld op <i>True</i> .

Tabel 3.8: Meegegeven parameters bij HuggingFace-requests

Naam	Prompt
P1	Vereenvoudig deze tekst
P2	Vereenvoudig deze tekst voor studenten (16-18 jaar) door moeilijke woorden te vervangen, vakjargon te schrappen, woorden langer dan 18 letters te vervangen, acroniemen voluit te schrijven, een woord slechts eenmaal door een synoniem te vervangen, korte uitleg te geven wanneer dat nodig is, en percentages te vervangen.
P3	Vereenvoudig een tekst door deze op te delen in kortere zinnen van maximaal tien woorden. Verander voornaamwoorden als 'zij', 'hun' of 'hij' in namen. Vervang complexe zinsconstructies en voorzetzes-zinnen door eenvoudiger alternatieven, maar laat ze ongewijzigd als er geen eenvoudiger optie beschikbaar is.

Tabel 3.9: De GPT-3-prompts die in de vergelijkende studie aan bod komen.

```
1 folder_path = 'scripts\\pdf'
2 dutch_spacy_model = "nl_core_news_md"
3 english_spacy_model = "en_core_web_sm"
4
5 dict = {
6     'nl':'nl_core_news_md',
7     'en':'en_core_web_sm'
8 }
9
10 total_df = None
11 gt = Translator()
12
13 huggingfacemodels = {
14     'T1':"https://api-inference.huggingface.co/models/haining/
15 scientific_abstract_simplification",
16     'T2': "https://api-inference.huggingface.co/models/sambydlo/bart-large-
17 scientific-lay-summarisation",
18     'T3': "https://api-inference.huggingface.co/models/philippeablan/
19 keep_it_simple"
20 }
21
22 max_length = 2000
23 COMPLETIONS_MODEL = "text-davinci-003"
24 EMBEDDING_MODEL = "text-embedding-ada-002"
25
26 languages = {
27     'nl':'nl_core_news_md',
28     'en':'en_core_web_md'
29 }
30
```

```
28 class HuggingFaceModels:
29     def __init__(self, key=None):
30         global huggingface_api_key
31         try:
32             huggingface_api_key = key
33         except:
34             huggingface_api_key = 'not_submitted'
35
36     def query(self, payload, API_URL):
37         headers = {"Authorization": f"Bearer {huggingface_api_key}"}
38         response = requests.post(API_URL, headers=headers, json=payload)
39         return response.json()
40
41     def scientific_simplify(self, text, lm_key):
42         try:
43             API_URL = huggingfacemodels.get(lm_key)
44             translated = GoogleTranslator(source='auto', target='en').translate(str(text))
45
46             if lm_key == 'T1':
47                 result = self.query({"inputs": str('simplify: ' + str(translated)), "parameters": {"max_length": len(sentence)+10}, "options": {"wait_for_model": True}}, API_URL)
48             else:
49                 result = self.query({"inputs": str(translated), "parameters": {"max_length": len(sentence)+10}, "options": {"wait_for_model": True}}, API_URL)
50
51             if 'generated_text' in result[0]:
52                 translated = GoogleTranslator(source='auto', target='nl').translate(str(result[0]['generated_text']))
53                 return translated
54             elif 'summary_text' in result[0]:
55                 translated = GoogleTranslator(source='auto', target='nl').translate(str(result[0]['summary_text']))
56                 return translated
57             else:
58                 return None
59         except:
60             return text
61
62     def get_sentence_length(sentence):
63         txt_language = detect(sentence)
64         dic_language = languages.get(txt_language)
65         nlp = spacy.load(dic_language)
66         doc = nlp(sentence)
67         return len()
68
69     def tokenize_text(text):
70         txt_language = detect(text)
71         dic_language = languages.get(txt_language)
```

```

73     nlp = spacy.load(dic_language)
74     doc = nlp(text)
75     return doc.sents
76
77
78 def process_file(file_path):
79     with open(folder_path + '/' + file_path, "r", encoding='utf8') as file:
80         text = file.read()
81         tokens = tokenize_text(text)
82         return tokens
83
84
85 hf = HuggingFaceModels(key=os.getenv('huggingface_key'))
86 original_scientific_papers = [f for f in os.listdir(folder_path)]
87
88 for paper in original_scientific_papers[3:]:
89     sentence_tokens = process_file(paper)
90     for sentence in sentence_tokens:
91         for model in huggingfacemodels.keys():
92             filename = "SIMPLIFIED_" + model + '_' + paper
93             with open(filename, 'a', encoding='utf-8') as f:
94                 output = hf.scientific_simplify(str(sentence), model)
95                 f.write(str(output))
96

```

Listing 3.3: Script voor de derde fase van de vergelijkende studie

Vervolgens berekent het script leesgraadscores met de *readability* python-bibliotheek. Leesgraadscores dienen, zoals aangegeven in Nenkova en Passonneau (2004), als objectieve maatstaf bij deze vergelijkende studie.

- De FRE en FOG zijn relevant, want ze kunnen de moeilijkheidsgraad van een zin of tekst objectief meten.
- Het aantal complexe en lange woorden kan wijzen op de gebruikte *substitution generation* van het taalmodel.
- Het aantal hulpwerkwoorden en vervoegingen van 'zijn' kan aanduiden op mogelijke passieve stem, wat het onderzoek van Ruelas Inzunza (2020) als 'hinderende' zinsyntax vernoemt.

Het resultaat van dit script is een *Pandas-dataframe* met alle leesbaarheidsmetrieken uit de *readability*-library. Ten slotte slaat het script de *Pandas-dataframe* op als CSV-bestand.

Listing 3.4 omvat de code om de leesmetrieken per zin bij te kunnen houden. Alleen eerst laadt het script de twee embeddingsmodellen, weergegeven in tabel 3.7 in. Vervolgens bouwt het script een leeg DataFrame op om de meetresultaten erin

op te slaan. Daarna itereert het python-script door alle teruggevonden tekstbestanden om vervolgens deze tekst in te lezen. Voor elke zin wordt geprobeerd om meetwaarden voor leesbaarheid te verkrijgen door gebruik te maken van de functie readability.getmeasures, waarbij de taal 'nl' (Nederlands) wordt gespecificeerd. Als het script alle meetwaarden succesvol kan verkrijgen, dan maakt het script een nieuwe rij met de objectieve leesmetrieken. Zo krijgt de dataframe alle leesmetrieken opgesomd in ?? . Ten slotte voegt het script alle rijen toe aan de DataFrame. Als er tijdens het meten van de leesbaarheid van een zin een Exception optreedt, dan verwerpt het script die zin voor dat artikel.

Leesgraadscore	Vereenvoudigingstechniek
FOG	LS en SS
FRE	LS en SS
Aantal woorden per zinnen	SS
Aantal complexe woorden per zin volgens Dale Chall index	LS
Aantal lange woorden per zin	LS
Aantal gebruikte hulpwerkwoorden	SS
Aantal zinnen met een vervoeging van het werkwoord 'zijn'	SS

Tabel 3.10: De objectieve metrieken voor de vergelijking van de vereenvoudigde teksten met de oorspronkelijke tekst en de referentieteksten.

```

1 simplified_folder = 'scripts/vereenvoudigde_artikelen'
2 original_folder = 'scripts/pdf'
3
4 scientific_papers = [original_folder + "/" + f for f in os.listdir(
5     original_folder)] + [simplified_folder + "/" + f for f in os.listdir(
6     simplified_folder)]
7
8 languages = {
9     'nl':'nl_core_news_md',
10    'en':'en_core_web_md'
11 }
12
13 df = pd.DataFrame()
14
15 for paper in scientific_papers:
16     with open(paper, 'r', encoding='utf-8') as file:
17         text = file.read()
18         nlp = spacy.load(languages.get('nl'))

```

```
17 doc = nlp(text)
18
19 for sent in doc.sents:
20     try:
21         metrics = readability.getmeasures(sent.text, lang='nl')
22         row = {
23             'Paper': paper.split('/')[-1].split('.')[0],
24             'Sentence': sent.text,
25             'FRE': metrics['readability grades']['FleschReadingEase'],
26             'FOG': metrics['readability grades']['GunningFogIndex'],
27         }
28
29         for key, value in metrics['sentence info'].items():
30             row[key] = value
31
32         for key, value in metrics['word usage'].items():
33             row[key] = value
34
35         for key, value in metrics['sentence beginnings'].items():
36             row[key] = value
37
38         df = df.append(row, ignore_index=True)
39     except Exception as e:
40         print(e)
41
42 df.to_csv('result.csv', index=False)
43
```

Listing 3.4: Script voor fase 4 van de vergelijkende studie

Tenslotte komen de resultaten van de menselijke beoordeling aan bod. Deze fase van de vergelijkende studie staat stil bij aspecten die leesmetrieken niet kunnen meten, waaronder de normen vermeld in tabel 3.11. De referentietekst dient hier als hulpmiddel om de referentietekst, ofwel het verwachte resultaat, te vergelijken met de vereenvoudigde tekst door een taalmodel.

Metriek	Vereenvoudigingstechniek
Acroniemen behouden	Lexicale vereenvoudiging
Inschatting van de doelgroep	Lexicale vereenvoudiging
Behoud van kern- en bijzaken	Lexicale vereenvoudiging
Schrijven in tabelvorm of als opsomming	Structurele aanpassing
Passieve zinconstructies herschrijven naar actieve zinconstructies	Syntactische vereenvoudiging
Bronvermelding behouden	N.v.t.
Citeren en parafraseren	N.v.t.

Tabel 3.11: Criteria voor menselijke observatie bij de vergelijkende studie.

3.3. Prototype voor tekstvereenvoudiging

Met het Moscow-schema en het geschikte taalmodel voor gepersonaliseerde ATS, kan het onderzoek een volgende stap zetten richting het beantwoorden van de onderzoeksvraag. Deze sectie omschrijft de ontwikkeling van een prototype voor gepersonaliseerde ATS voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Daarmee kan de ontwikkeling een antwoord bieden op de volgende deelvraag:

- Hoe kan een intuïtieve en lokale webtoepassing worden ontwikkeld die zowel scholieren met dyslexie als docenten helpt bij het vereenvoudigen van wetenschappelijke artikelen met behoud van semantiek, jargon en zinsstructuren?

Voor de ontwikkeling van het prototype volgt het onderzoek de flowchart op figuur 3.3. Deze flowchart toont zes algemene fasen. Zo start het onderzoek met een voorbereidende fase waarin het onderzoek de nodige technieken en taalmodellen opsomt. Vervolgens start het onderzoek met de ontwikkeling van de back end en front end. Vervolgens komt de ontwikkeling van het lerarencomponent, gevolgd door de ontwikkeling van het scholierencomponent aan bod. Daarna moet het onderzoek stilstaan bij de opzet van het prototype met behulp van Docker. Ten slotte evalueert het onderzoek het gemaakte prototype aan de hand van het Moscow-schema en de vergelijkende studie. Het lerarencomponent moet wetenschappelijke artikelen kunnen vereenvoudigen, na selectie van gepersonaliseerde ATS-technieken. Daarnaast moet het scholierencomponent ondersteuning bieden. In dit component van het prototype moeten scholieren met dyslexie in *real-time* aanpassingen kunnen maken aan de tekst, alsook ondersteuning krijgen tijdens het begrijpend lezen van een tekst. Gepersonaliseerde opmaakopties moeten over

de volledige prototype gelijk blijven.

Taalmodellen selecteren

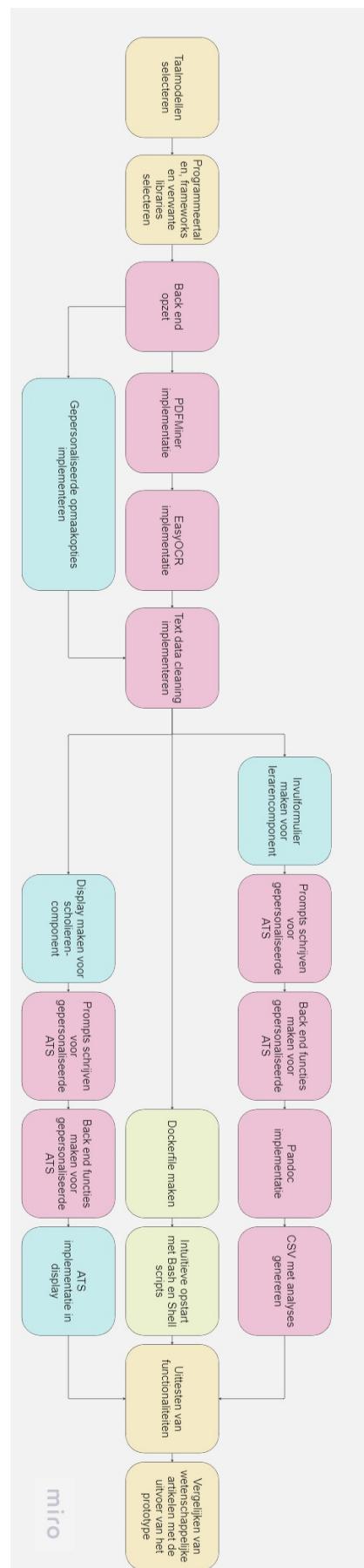
Allereerst bepaalt het onderzoek een taalmodel of meerdere taalmodellen voor ge-personaliseerde ATS, omdat deze keuze later de technologiekeuze kan beïnvloeden. Zo wijst het onderzoek uit dat GPT-3 een geschikt taalmodel is voor gepersonaliseerde ATS. Na de bepaling van het taalmodel bepaalt het onderzoek de technologieën en frameworks voor de *back end* en de *front end*.

Programmeertalen, frameworks en verwante libraries selecteren

Omwille van de beschikbaarheid van GPT-3 via API-calls, kunnen python en javascript geschikte keuzes voor dit prototype. Voor een snelle en gratis ontwikkeling gebruikt het prototype *open-source* pakketten. Tabel 3.12 toont een breed overzicht van alle gebruikte programmeertalen. Hierop vult tabel 3.13 aan door een overzicht te geven van alle gebruikte python-libraries.

Technologie	Functionaliteit
Python	De back end van het prototype die API-calls en de NLP-functionaliteiten zoals PoS-tagging en lemmatizing verwerkt.
JavaScript (JS)	De toepassing gebruikersvriendelijker maken, personalisatie-opties voor de site doorvoeren en de functies gebouwd in Javascript dienen als alternatief op commandline instructies.
HTML en CSS	Het visuele uiterlijk van de website aanpassen naargelang de gekozen parameters van de eindgebruiker.
Jinja	Informatie uit de back end doorgeven aan de front-end.
Docker	Lokale uitrol van de webtoepassing.
Bash	Intuïtief script om de webtoepassing op te starten voor Linux en Mac-systemen.
Powershell	Intuïtief script om de webtoepassing op te starten op Windows-systemen.

Tabel 3.12: Gebruikte programmeertalen in het prototype voor tekstvereenvoudiging.

**Figuur (3.3)**

Algemeen overzicht van de ontwikkeling van het prototype voor ATS van wetenschappelijke artikelen.

Python-bibliotheek	Functionaliteit
Flask	Het framework van de webtoepassing. Dit framework combineert front end en back end.
PDFMiner	Tekstinhoud van PDF's extraheren.
EasyOCR	PDF-pagina's inscannen als afbeelding in JPG-formaat om vervolgens de tekst te extraheren.
NumPy	De reshape-functie vereenvoudigt de manier om arrays van zinnen bij elkaar te plaatsen om zo een paragraaf te bekomen.
Spacy	PoS-tagging en het lemmatiseren van woorden.
OpenAI	GPT-3 API aanspreken.
BeautifulSoup	HTML-content in het lerarencomponent <i>parsen</i> voor een correcte formatting.

Tabel 3.13: Gebruikte Python-libraries en hun respectievelijke functie in het prototype.

Back end opzet

Na het kiezen van de taalmodellen en de technologieën en frameworks, start het onderzoek met de front end implementatie. Allereerst voegt het onderzoek een homepagina toe dat als portaal zal dienen voor de leraren- en scholierencomponenten. Allereerst implementeert het onderzoek de nodige personaliseerbare opmaakopties. Zoals aangeraden door Galliussi e.a. (2020), moeten ontwikkelaars rekening houden met de opmaakopties van de website. Zo maakt het prototype gebruik van alle parameters onderzocht door Rello en Baeza-Yates (2013) en Rello, Baeza-Yates, Dempere-Marco e.a. (2013). Het instellingenscherm is een HTML-bestand waarin een formulier alle aanpasbare parameters oplijst. Tabel ... toont de opgenomen parameters. De standaardparameters volgen de aangeraden parameters van (...). Het formulier stuurt vervolgens een POST-request naar de back end. De back end verwerkt deze aanvraag en slaat de ontvangen dictionary op als sessievariable, zoals weergegeven in listing 3.5.

```

1 PER_SET_SESSION_NAME = 'personalized_settings'
2
3 @app.route('/get-settings-user', methods=['POST'])
4 def return_personal_settings_dict():
5     if PER_SET_SESSION_NAME in session:
6         return jsonify(session[PER_SET_SESSION_NAME])
7     else:
8         return jsonify(result='session does not exist')
9
10 @app.route('/change-settings-user', methods=['POST'])

```

```

11 def change_personal_settings():
12     try:
13         session[PER_SET_SESSION_NAME] = dict(request.form)
14         msg = 'Succesvol aangepast!'
15     except Exception as e:
16         msg = str(e)
17         flash(msg)
18     return render_template('settings.html')

```

Listing 3.5: De back end functie die de aanpassingen uit het formulier opslaat als sessievariabele.

Ten slotte moet de *front end* deze opmaakopties bij iedere wegpagina inladen. Het inladen gebeurt met een *window onload*, zoals weergegeven in listing 3.6. Zo hoeven eindgebruikers deze parameters niet bij iedere webpagina opnieuw aan te passen.

```

1 window.onload = async function () {
2     var url = `http://localhost:5000/get-settings-user`;
3     const response = await fetch(url, { method: 'POST' });
4     var result = await response.json();
5     document.body.style.fontSize = result.fontSize+'px';
6     document.body.style.fontFamily = result.fontSettings;
7     document.body.style.backgroundColor = result.favcolor;
8     document.body.style.lineHeight = result.lineHeight+'em';
9     document.body.style.wordSpacing = result.wordSpacing+'em';
10    document.body.style.textAlign = result.TextAlign;
11 }

```

Listing 3.6: De onload-functie die de gepersonaliseerde opmaakopties regelt bij het inladen van een webpagina.

Naast de gepersonaliseerde opmaakopties, moet het prototype ook over een systeem beschikken dat wetenschappelijke manieren kan opladen en inlezen. Zo biedt het prototype twee manieren aan om wetenschappelijke artikelen in te laten: via *plaintext* of via een PDF-bestand. Niet alle *PDF extractors* zijn foutbestendig, want zoals eerder opgemerkt in de 3.1 kunnen *PDF extractors* ook tekst verliezen tijdens dit proces. Als valnet biedt het prototype een OCR-optie aan. Deze functionaliteit omvat de ontwikkeling van functionaliteiten aan de front-end en aan de back end.

De *front end* kent enkel de toevoeging van een checkbox en twee *input-tags*, namelijk een *file-input* en een *textarea-input*. De *file-input* geeft de binaries van het PDF-bestand mee van de front-end naar de back end die het PDF-bestand tijdelijk *in-memory* opslaat. Daarnaast krijgt de back end de keuze van de gebruiker tussen normale en OCR-upload mee als boolean, waarbij 1 gelijk staat aan een OCR-afhandeling. Ten slotte gebruikt het formulier een POST-request om alle meegekregen informatie door te sturen naar de back end. Na het ontvangen van de request van de front-end, handelt de Flask back end het bestand verder af zoals aan-

gewezen in listing 3.7. Eerst controleert de back end het type invoer. Vervolgens spreekt de Flask back end de Reader-klasse aan die de tekst formatteert tot een bruikbaar formaat voor de webtoepassing. Daarna gebruikt de Reader-klasse één van twee mogelijke technieken.

```
1 def setup_scholars_teachers(request):
2     settings = request.form
3     if 'fullText' in request.form:
4         text = request.form['fullText']
5         langs = detect_langs(text)
6         reader = Reader()
7         dict_text = reader.get_full_text_site(text)
8     elif 'pdf' in request.files:
9         if 'advanced' not in settings:
10            pdf = request.files['pdf']
11            pdf_data = BytesIO(pdf.read())
12            all_pages = extract_pages(pdf_data, page_numbers=None, maxpages=999)
13            langs = detect_langs(str(all_pages))
14            reader = Reader()
15            full_text = reader.get_full_text_dict(all_pages)
16            dict_text = reader.get_full_text_site(full_text)
17        else:
18            pdf = request.files['pdf']
19            pdf_data = pdf.read()
20            pages = convert_from_bytes(pdf_data)
21            reader = Reader()
22            img_text = reader.get_full_text_from_image(pages)
23            langs = detect_langs(img_text)
24            dict_text = reader.get_full_text_site(img_text)
25        return dict_text, langs, 'voorbeeldtitel', 'voorbeeldonderwerp'
26
27 @app.route('/for-scholars', methods=['GET', 'POST'])
28 def teaching_tool():
29     try:
30         dict_text, langs, title, subject = setup_scholars_teachers(request)
31         return render_template('for-scholars.html', pdf=dict_text, lang=langs, title=title, subject=subject)
32     except Exception as e:
33         return render_template('error.html', error=str(e))
34
35 @app.route('/for-teachers', methods=['GET', 'POST'])
36 def analysing_choosing_for_teachers():
37     try:
38         dict_text, langs, title, subject = setup_scholars_teachers(request)
39         return render_template('for-teachers.html', pdf=dict_text, lang=langs, title=title, subject=subject)
40     except Exception as e:
41         return render_template('error.html', error=str(e))
```

Listing 3.7: Koppeling tussen front-end en back-end voor het inlezen van een wetenschappelijk artikel

PDFMiner implementatie

Bij een gewone PDF-extractie gebruikt de Reader-klasse PDFMiner met de functie in listing 3.8. Deze functie itereert doorheen alle PDF-pagina's van een meegekregen *in-memory* pdf. Nadien extraheert de functie alle tekst op een pagina. Nadien concateneert deze functie alle opgehaalde tekst van één pagina aan een lege string. De functie herhaalt dit proces tot het script geen pagina's meer kan inlezen. Het resultaat van deze functie is een string-object met alle geëxtraheerde tekst uit een pdf.

```

1 def get_full_text_from_pdf(self, all_pages):
2     total = ""
3     for page_layout in all_pages:
4         for element in page_layout:
5             if isinstance(element, LTTextContainer):
6                 for text_line in element:
7                     total += text_line.get_text()
8     return total

```

Listing 3.8: Een PDF inlezen met PDFMiner

EasyOCR implementatie

PDFMiner kan tekst missen bij de pdf-extractie. Daarom voorziet het prototype een valnet met een OCR-techniek. De Python-bibliotheek EasyOCR voorziet een eenduidige en ontwikkelaarsvriendelijke manier om tekst uit pdf's te extraheren met OCR-technieken. Zo itereert EasyOCR over alle pagina's in een PDF en slaat iedere pagina op als een JPG-bestand. Vervolgens leest EasyOCR de tekst op iedere pagina in en houdt dit bij tot het einde van het script. Na het inlezen van de tekst, verwijdert het script de aangemaakte afbeeldingen om zo geheugenruimte te besparen. Net zoals bij de eerste methode, resulteert deze methode in een string-object met alle tekst uit de PDF. De uitgewerkte code staat uitgeschreven in listing 3.9

```

1 def get_full_text_from_image(self, all_pages):
2     img_files = []
3     num_pages = 0
4     for i, page in enumerate(all_pages):
5         file = f'page_{num_pages}.jpg'
6         page.save(file, 'JPEG')
7         img_files.append(file)
8         num_pages += 1
9
10    full_text = []
11    reader = easyocr.Reader(['nl'])
12    for f in img_files:
13        result = reader.readtext(f, detail=0)
14        full_text.append(" ".join(result))
15        os.remove(f)
16    return " ".join(full_text)

```

Listing 3.9: Een PDF inlezen met OCR

Geëxtraheerde inhoud formatteren naar een webpagina.

Na het extraheren van de tekstinhoud, komt een formatteerfase aan bod. Hier vormt het systeem de tekst om naar het gewenste formaat voor de front end. Allereerst transformeert de back end de string van geëxtraheerde tekst naar arrays van zinnen met Spacy *word embeddings* en *sentence embeddings*. Deze functie staat beschreven in listing 3.10. Het prototype bundelt vijf zinnen met de *reshape*-functie van Numpy. Eindgebruikers kunnen deze parameter aanpassen in het instellingenscherm. Om de PoS-tag bij het respectievelijke woord bij te houden, gebruikt het prototype een *key-value pair* datastructuur. Zo verwijst de sleutel naar een woord in een zin en de waarde verwijst naar de PoS-tag. Het prototype houdt rekening met enkel Nederlandstalige en Engelstalige wetenschappelijke artikelen. Daarom laadt het prototype hoogstens twee embeddingsmodellen op. Deze embeddingsmodellen staan vermeld in tabel 3.7. Het prototype vermijdt hardcoderen. Daarom maakt het gebruik van een *dictionary* die de naam van deze embeddingsmodellen bijhoudt. Het prototype gebruikt Engels als standaardtaal, als de back end de taal niet kan herkennen of als de taal niet in de lijst van de dictionary staat.

```
1 def get_full_text_site(self, full_text):
2     try:
3         lang = detect(full_text)
4     except:
5         lang = 'en'
6
7     if lang in dict:
8         nlp = spacy.load(dict.get(lang))
9     else:
10        nlp = spacy.load(dict.get('en'))
11
12    full_text = str(full_text).replace('\n', ' ')
13
14    doc = nlp(full_text)
15    sentences = []
16    for sentence in doc.sents:
17        sentences.append(sentence)
18
19    pad_size = SENTENCES_PER_PARAGRAPH - (len(sentences) % SENTENCES_PER_PARAGRAPH)
20    padded_a = np.pad(sentences, (0, pad_size), mode='empty')
21    paragraphs = padded_a.reshape(-1, SENTENCES_PER_PARAGRAPH)
22
23    text_w_pos = []
24    for paragraph in paragraphs:
25        paragraph_w_pos = []
26        try:
27            for sentence in paragraph:
```

```

28     dict_sentence = {}
29     for token in sentence:
30         dict_sentence[token.text] = str(token.pos_).lower()
31     paragraph_w_pos.append(dict_sentence)
32     text_w_pos.append(paragraph_w_pos)
33     except:
34         pass
35     return text_w_pos

```

Listing 3.10: Het formatteren van de tekst naar een formaat voor de website.

Zowel het leraren- als het scholierencomponent passen deze dynamische HTML-structuur in hun weergave toe. Zo zijn er vier verschillende klassen die aan span-tags in deze weergave worden toegekend. Door middel van een Jinja-iteratie, krijgt iedere *span-tag* een specifieke klasse afhankelijk van de key of iteratie. Deze iteratie toont listing 3.11 voor. Zo is er een overkoepelende span-tag *sentence*, alsook voor ieder woord in een zin hoort er een span-tag met *nouns*, *adjectives* of *verbs*. Andere woorden, zoals conjuncties, behoren tot de klasse *other*. Na het extraheren van de teksthoud van een wetenschappelijk artikel, komt de ontwikkeling van het leraren- en scholierencomponent aan bod. Hoewel ontwikkelaars de ontwikkeling van beide componenten parallel kunnen laten lopen, start het onderzoek met de werkwijze voor de ontwikkeling van het lerarencomponent.

```

1  {% for paragraph in pdf %}
2      <p class="left-side">
3          {% for sentence in paragraph %}
4              <span class="sentence">
5                  {% for word in sentence %}
6                      {% if sentence[word] != 'space' %}
7                          <span class="{{sentence[word]}}>{{word}}</span>
8                      {% endif %}
9                  {% endfor %}
10             </span>
11         {% endfor %}
12     </p>
13 {% endfor %}

```

Listing 3.11: Het doorlopen van de PDF-tekst op de webpagina en het toekennen van de span-tags.

Invulformulier maken voor het lerarencomponent.

Zo kunnen leerkrachten beschikken over een tool waarin zij de geëxtraheerde teksthoud kunnen manipuleren, om vervolgens opties voor gepersonaliseerde ATS te selecteren. Figuur 4.17 toont een mogelijke weergave van deze HTML-pagina. Leerkrachten moeten in dit lerarencomponent kunnen beschikken over de functionaliteiten weergegeven in tabel 3.14. Tabel 3.2 reikt de benodigde opties voor gepersonaliseerde ATS aan. Het prototype gebruikt deze criteria als opties om de prompts dynamisch op te bouwen.

Functionaliteit	Gebruikte JS of python-techniek
Specifieke prompt meegeven per paragraaf	Naast een optie om voor het hele document één prompt te gebruiken, voegt het prototype ook een optie toe om. Hiervoor past de web-inhoud een key-value structuur toe.
Opties voor gepersonaliseerde ATS aanreiken.	Met behulp van een HTML-formulier kunnen leerkrachten opties aanvinken waaraan de vereenvoudigde tekst moet voldoen.
Werkwoorden, bijvoeglijke en zelfstandige naamwoorden markeren	Front-end aanpassing met <i>eventlistener</i> . De tekstkleur van het aangeduide type woorden verandert naar het gekozen kleur.
Zinnen verwijderen	Front-end filter aangesproken door een <i>eventlistener</i> .
Woord toevoegen aan de woordentlijst	Een <i>eventlistener</i> handelt de functionaliteit af door de woorden en hun zin van voorkomen tijdelijk op te slaan in een storage. Het formulier houdt dit bij en geeft het vervolgens mee bij het indienen. Deze woorden en hun respectievelijke zin van voorkomen dienen om de woordentlijst op te vullen in het gegenereerde PDF of Word-bestand.
Sleutel en sessieherkenning	.

Tabel 3.14: Alle beschikbare functionaliteiten in

Prompts schrijven voor gepersonaliseerde ATS in het lerarencomponent

Het onderzoek stelt de prompts op volgens de richtlijnen beschreven in tabel 2.12. Daarnaast past het Intent- en Contextualization prompts toe zoals aangeraden door White e.a. (2023). Tabel 3.15 geeft een overzicht van alle opgestelde prompts die het prototype gebruikt in het lerarencomponent. Daarnaast gebruikt iedere API-call de parameters omschreven in tabel 3.16.

Functionaliteit	Prompt
Synoniem opzoeken	Geef een eenvoudiger synoniem voor 'woord'. Context context.
Definitie opzoeken	Geef een eenvoudige definitie voor 'woord'. Context context.
Zin vereenvoudigen	
Gepersonaliseerde ATS	
Gepersonaliseerde ATS als opsomming	Herschrijf dit als een lijst van vereenvoudigde zinnen met (gepersonaliseerde opties) :return: een lijst van vereenvoudigde zinnen gesplitst door een ' ' teken /// context

Tabel 3.15: Tabel met de gebruikte prompts voor het lerarencomponent.

Parameter	Gebruikte waarde
Prompt	Zie tabel 3.15
Temperature	0
Max tokens	De lengte van het woord vermeerdert met tien tokens als het over het opzoeken van een definitie of synoniem gaat. Of de lengte van de zin vermeerdert met twintig tokens indien het over vereenvoudiging van een zin gaat.
Model	Da Vinci 3
Top-p	90%
Stream	False

Tabel 3.16: Gebruikte parameters om de definitie van een woord te genereren met GPT-3.

Back end functies maken voor gepersonaliseerde ATS

Leraren kunnen handmatig zinnen uit de tekst verwijderen. Hiervoor gebruikt de *front end* een *eventlistener* die de aangeklikte span-tag van klasse 'sentence' uit het HTML-document verwijdert. Naast zinnen verwijderen kunnen leraren ook woorden aan een woordenlijst toevoegen. Hiervoor gebruikt de front-end een JS-script met een *eventlistener* die een woord en de toebehorende zin toevoegt aan een *textarea*. Deze *textarea* gebruikt een *pipe*-symbool als separator om nadien de text te splitsen en een array te bekomen. Om de leerkracht opties aan te reiken

voor gepersonaliseerde ATS, bevat het HTML-document een formulier met keuzes aan. Dit formulier bevat alle nodige MTS-technieken uit tabel 3.2. Na een POST-request krijgt de back end deze parameters die het systeem nadien verwerkt in de prompt voor het GPT-3 taalmodel, zoals aangewezen in listing ???. Leraren kunnen ook het uitvoerbestand personaliseren met lettertypes, gekozen regeleinde, woordspatiëring en de marge van het document. Nadat de back end de inhoud van dit formulier ontvangt, neemt de GPT-klasse de aanvraag over. Deze klasse verwerkt drie functionaliteiten, namelijk definities en synoniemen opzoeken en gepersonaliseerde ATS.

Allereerst dient de *look-up* functie om de definitie van een woord te achterhalen. Zo beschrijft listing 3.12 de functie om met GPT-3 een eenvoudige definitie van een woord te genereren. Om de uitvoer zo waarschijnlijk mogelijk te maken, gebruikt de GPT-3 API-call een *nultemperature*. Daarna haalt het script het verkregen antwoord uit de JSON-response om deze vervolgens tijdelijk bij te houden in een dictionary.

```

1 def look_up_word_gpt(self, word, context):
2     try:
3         prompt = f"""
4             Give a simple Dutch explanation in one sentence for this word in the given
5             context. Give the PoS-tag and Dutch definition: '{word}'
6             context: {context}
7             format: PoS-tag | definition
8             """
9
10        result = openai.Completion.create(
11            prompt=prompt,
12            temperature=0,
13            max_tokens=50,
14            model=COMPLETIONS_MODEL,
15            top_p=0.9,
16            stream=False
17        )["choices"][0]["text"].strip("\n")
18        return result, word, prompt
19    except Exception as e:
20        return 'error', str(e), str(e)

```

Listing 3.12: Een functie om met GPT-3 een vereenvoudigde definitie voor een woord te genereren.

Vervolgens gebruikt de back end het GPT-3 model om een synoniem op te zoeken. Hiervoor spreekt de API de functie *give-synonym* aan, verwezen in listing ???.

```

1     """ @returns prompt, result from gpt """
2     def give_synonym(self, word, context):
3         try:
4             prompt = f"""
5                 Give a Dutch synonym for '{word}'. If there is no Dutch synonym available,
6                 explain it between curly brackets.

```

```

6 context:
7 {context}
8 """
9
10 result = openai.Completion.create(
11 prompt=prompt,
12 temperature=0,
13 max_tokens=10,
14 model=COMPLETIONS_MODEL,
15 top_p=0.9,
16 stream=False
17 )["choices"][0]["text"].strip("\n")
18 return result, word, prompt
19 except Exception as e:
20     return 'Open AI outage or problemen', str(e)

```

Listing 3.13: Een synoniem genereren of ophalen met GPT-3.

Daarna komt de vereenvoudiging van zinnen aan bod. Listing 3.14 bevat de code die deze API-call afhandelt. Echter moet het script rekening houden met twee aspecten. Allereerst kan de leerkracht kiezen voor een opsomming van de tekst. Daarnaast moet de inputlengte overeenstemmen met de beperkingen van de GPT-3 API. Wetenschappelijke artikelen zijn bijna altijd langer dan de mogelijk inputlengte. Daarom splitst het prototype de oorspronkelijke tekst op per 1000 tokens, zodat de inputprompt over marge beschikt en daardoor alle nodige gepersonaliseerde ATS-opties mee kan geven. Bij deze splitsing houdt het script rekening met de volledigheid van een zin.

```

1 def personalised_simplify(self, sentence, personalisation):
2     if 'summary' in personalisation:
3         prompt = f"""
4             Simplify the sentences in the given text and {", ".join(personalisation)}
5             :return: A list of simplified sentences divided by a '|' sign
6             /**
7             {sentence}
8             """
9     else:
10        prompt = f"""
11            Explain this in own Dutch words and {", ".join(personalisation)}
12            /**
13            {sentence}
14            """
15
16    try:
17        result = openai.Completion.create(
18            prompt=prompt,
19            temperature=0,
20            max_tokens=len(prompt),
21            model=COMPLETIONS_MODEL,
22            top_p=0.9,

```

```

23     stream=False
24 )["choices"][0]["text"].strip("\n")
25
26 if 'summary' in personalisation:
27     result = result.split('|')
28 else:
29     result = [result]
30
31 return result, prompt
32 except Exception as e:
33     return str(e), prompt

```

Listing 3.14: Een zin gepersonaliseerd vereenvoudigen met GPT-3.

Pandoc implementatie

Ten slotte giet het script de *plain-text* van vereenvoudigde tekstinhoud en de woordenlijsten in een PDF- of DOCX-bestand. Zo maakt de Creator-klasse PDF's en docx-documenten volgens de meegegeven opmaakopties en maakt gebruik van Pandoc via python. Pandoc gebruikt een tweestapsbeweging, waarbij het eerst *plain-text* naar een markdownformaat omzet en vervolgens het Markdown-bestand naar een PDF of Word-document converteert. Daarvoor is een YAML-header nodig die de elementen, beschreven in tabel 3.17, moet bevatten. Het script in listing ?? voegt de meegekregen instellingen toe aan de YAML-header.

Label in YAML-header	Voorbeeldwaarde
Title	Surveillance met artificiële intelligentie.
Mainfont	Arial
Titlefont	Arial Black
Date	14-06-2023
Document	Article
Margin	3cm
Word-spacing	0.3cm
Lineheight	singleheight

Tabel 3.17: Benodigde labels voor een gepersonaliseerd document met Pandoc.

Listing 3.15 bouwt de YAML-header op voor het markdownbestand. Deze header is volledig parameteriseerbaar.

```

1 markdown_file = "saved_files/file.md"
2 DATE_NOW = str(date.today())
3
4 class Creator():
5     def create_header(self, title, margin, fontsize, chosen_font, chosen_title_font,
                      word_spacing, type_spacing):

```

```

6     with open(markdown_file, 'w', encoding='utf-8') as f:
7         f.write("---\n")
8         f.write(f"---\n")
9         f.write(f"title: {title}\n")
10        f.write(f"mainfont: {chosen_font}.ttf\n")
11        f.write(f"titlefont: {chosen_title_font}.ttf\n")
12        f.write(f'date: {DATE_NOW}\n')
13        f.write(f'document: article\n')
14        f.write(f'geometry: margin={margin}cm\n')
15        f.write(f'fontsize: {fontsize}pt\n')
16        f.write('header-includes:\n')
17        f.write(f'- \spaceskip={word_spacing}cm\n')
18        f.write(f'- \\usepackage{{setspace}}\n')
19        f.write(f'- \\{type_spacing}\n')
20        f.write("---\n")

```

Listing 3.15: Writer-klasse omvattende de code om dynamische PDF- en Word-documenten te genereren.

Om de woordenlijst aan het markdown-bestand toe te voegen, bouwt het script eerst een *dictionary*-structuur op met de positie van het woord als key en als values de woord, de PoS-tag en de opgehaalde gepersonaliseerde betekenis. Het prototype moet rekening houden met homoniemen en daarom kan de key hier niet het woord zijn. Bij een lege woordenlijst komen deze bewerkingen niet aan bod.

```

1 def generate\_glossary(self, list):
2 with open(markdown_file, 'a', encoding='utf-8') as f:
3     f.write(" ---\n")
4     f.write("# Woordenlijst\n")
5     f.write("| Woord | Soort | Definitie |\n")
6     f.write("| --- | --- | --- |\n")
7     for word in list.keys():
8         f.write(f"| {word} | {list[word]['type']} | {list[word]['definition']} |\n")

```

Listing 3.16: Een woordenlijst genereren met de Writer-klasse.

Deze functie vult het markdown-bestand met de vereenvoudigde tekst, gesplitst door de titels die de leerkracht heeft gekozen. Het script slaat de vereenvoudigde tekst nadien op in een *dictionary*-structuur. Vervolgens print het script de vereenvoudigde tekst uit naar het markdown-bestand door alle titels van de *dictionary*-structuur te doorlopen. Een titel uitprinten in markdown syntax moet voorafgaan aan twee *hashtags*, gevolgd door een *breakline*.

```

1 def generate_simplification(self, full_text):
2     with open(markdown_file, 'a', encoding="utf-8", errors="surrogateescape") as f:
3         for key in full_text.keys():
4             title = str(key).replace('\n', ' ')
5             text = full_text[key]
6             f.write('\n\n')
7             f.write(f'## {title}')
8             f.write('\n\n')

```

```

9 f.write(" ".join(text))
10 f.write('\n\n')

```

Listing 3.17: Een doorlopende tekst toevoegen aan het markdownbestand met de Writer-klasse.

Indien de leerkracht een opsomming wenste, dan dient de titel nog steeds al separator, maar het script print de zinnen uit als opsomming conform aan de Markdown-syntax. Na de titel print het script de vereenvoudigde tekst per paragraaf uit. Bij een opsomming gaat een asterisk-symbool vooraf. Vervolgens converteert Pandoc het Markdown-bestand naar een PDF-bestand gebouwd met de XeLaTeX engine of een Word-bestand met de meegekregen binaries.

```

1 def generate_simplification_w_summation(self, full_text):
2     with open(markdown_file, 'a', encoding="utf-8", errors="surrogateescape") as f:
3         for key in full_text.keys():
4             title = str(key).replace('\n', ' ')
5             text = full_text[key][0].split('|')
6             f.write('\n\n')
7             f.write(f'## {title}')
8             for sentence in text:
9                 f.write('\n\n')
10            f.write(f'* {sentence}')
11            f.write('\n\n')

```

Listing 3.18: Een opsomming toevoegen aan het markdownbestand met de Writer-klasse.

Tenslotte maakt het script de pdf en docx-bestanden van de vereenvoudigde teksten. De functie verwezen in listing 3.19 gebruikt alle bovenstaande (nodige) functies. Daarna genereert Pandoc de pdf en docx-bestanden. Nadien zal de functie deze twee bestanden comprimeren tot één zip-bestand. Hoewel Flask maar één bestand kan teruggeven, comprimeert het script met de `zipfile` bibliotheek deze twee bestanden tot één bestand. Zo krijgt de eindgebruiker alsnog zowel het docx als het pdf-document.

```

1 def create_pdf(self, title, margin, list, full_text, fonts, word_spacing,
               type_spacing, summation):
2     if title is not None:
3         self.create_header(title=title, margin=margin, fontsize=14, chosen_font=fonts[0],
4                            chosen_title_font=fonts[1], word_spacing=word_spacing, type_spacing=
4                            type_spacing)
4     else:
5         self.create_header(title='Vereenvoudigde tekst', margin=0.5, fontsize=14,
6                            chosen_font=fonts[0], chosen_title_font=fonts[1], word_spacing=word_spacing,
6                            type_spacing=type_spacing)
6
7     """GLOSSARY"""
8     if len(list) != 0:
9         self.generate_glossary(list=list)
10
11    """SUMMARY"""

```

Functionaliteit	JS/GPT-3
Zinnen verwijderen	Enkel JS
Zinnen vereenvoudigen met gepersonaliseerde keuzes	GPT-3 en JS
Zinnen vereenvoudigen met prompt	GPT-3 en JS
Woord aan woordenlijst toevoegen	GPT-3 en JS
Woorden (werkwoorden, bijvoeglijke en zelfstandige naamwoorden) markeren	JS

Tabel 3.18: Beschikbare functionaliteiten in het scholierencomponent.

```

12 if summation:
13     self.generate_summary_w_summation(full_text=full_text)
14 else:
15     self.generate_summary(full_text=full_text)
16
17 """FILE_CREATION"""
18 pypandoc.convert_file(source_file=markdown_file, to='docx', outputfile=docx_file,
19     extra_args=['-M2GB', '+RTS', '-K64m', '-RTS'])
20 pypandoc.convert_file(source_file=markdown_file, to='pdf', outputfile=pdf_file,
21     extra_args=['--pdf-engine=xelatex'])
22 with zipfile.ZipFile(zip_filename, 'w') as myzip:
23     myzip.write(pdf_file)
24     myzip.write(docx_file)

```

Listing 3.19: Een zip-bestand aanmaken met daarin een docx en pdf bestand van de vereenvoudigde tekst.

De functionaliteiten van het lerarencomponent stoppen hier. Vervolgens komt het scholierencomponent aan bod, waarbij de nadruk ligt op het ontwikkelen van een ondersteunende tool.

3.3.1. De ontwikkeling van het scholierencomponent.

Tabel 3.18 geeft een overzicht van alle functionaliteiten die in het scholierencomponent moeten zitten.

Display maken voor scholierencomponent

Net zoals bij het lerarencomponent, moet het prototype een oorspronkelijke weergave van het wetenschappelijk artikel kunnen tonen. Het onderzoek baseert de lay-out op dat van de erkende softwarepakketten, alsook de uitgeteste chatbots.

Prompts schrijven voor gepersonaliseerde ATS

De prompts in tabel 3.15 kan het scholierencomponent overnemen van het lerarencomponent. Aanvullend op deze prompts kunnen scholieren ook zelf een prompt schrijven.

Back end functies schrijven voor gepersonaliseerde ATS

Om zinnen in de doorlopende tekst te verwijderen, moet de front end over de nodige span-tags beschikken. Bij het inlezen van de PDF voert de back end dit al uit, zoals verwezen in listing 3.10. Daarom hoeft het onderzoek geen nieuwe functie in de back end te schrijven. Vervolgens moet de back end over een functie beschikken om een zin te vereenvoudigen. Het lerarencomponent gebruikt al dergelijk functie. Daarmee hergebruikt de back end de functie in listing 3.14.

Zinnen baseren op een zelfgemaakte prompt moet de back end echter wel toevoegen. Zo gebruikt de back end de functie in listing 3.20 om een *custom prompt* te verwerken. Deze functie stuurt de oorspronkelijke prompt, samen met de context, door naar de GPT-3 API.

```
1 def personalised_simplify_w_prompt(self, sentences, personalisation):
2     try:
3         result = openai.Completion.create(
4             prompt=personalisation,
5             temperature=0,
6             max_tokens=len(personalisation)+len(sentences),
7             model=COMPLETIONS_MODEL,
8             top_p=0.9,
9             stream=False
10        )["choices"][0]["text"].strip("\n")
11        return result, personalisation
12    except Exception as e:
13        return str(e), personalisation
```

Listing 3.20: Een API-call sturen naar GPT-3 met een custom prompt.

Vervolgens heeft de back end als een functie die een woord opzoekt met GPT-3. Deze functie staat beschreven in listing 3.12 en de back end kan deze functie zonder problemen hergebruiken. Ten slotte hoeft de back end geen nieuwe functie te krijgen om woordmarkering af te handelen.

Front end implementatie voor ATS functionaliteiten

Allereerst kan de front end zonder hulp van de back end zinnen met JS verwijderen. Alle woorden in een zin bundelt de front end in een span-tag van de klasse 'sentence'. Als de gebruiker dergelijk span-tag aanklikt, dan verwijdert de front end de gekozen span-tag.

Eerst slaat JS de gemarkeerde tekst en meegekregen ATS-opties op en geeft deze door aan de back end met een API-call. Vervolgens verwerkt de back end deze aanvraag door een nieuwe aanvraag te sturen naar GPT-3, met daarin een prompt die de tekst en de gekozen ATS-technieken bevat. De prompt specificeert het formaat waaraan de uitvoer van het taalmodel moet voldoen. Als de tekst doorlopend is, dan verwerkt JS dit resultaat als een p-tag. Als het taalmodel een opsomming

moest genereren, dan zal de front-end alle zinnen doorlopen en uitprinten tussen twee li-tags.

Listing 3.21 toont de aanpak om via de front end een woord, pos-tag en definitie toe te voegen aan de woordenlijst tabel. De front end handelt enkel aanvragen voor werkwoorden, adjektieven, hulpwerkwoorden en zelfsandige naamwoorden af. Eenmaal de scholier op een woord drukt, stuurt de front end een aanvraag naar de back end. Hiervoor stuurt de front end de context en het woord naar de back end. De front end voegt nadien een nieuwe rij aan de tabel toe met daar in het woord, de PoS-tag en de definitie.

```

1 document.addEventListener("DOMContentLoaded", () => {
2   const spans = document.querySelectorAll(".verb, .adj, .noun, .aux");
3   spans.forEach((span) => {
4     span.addEventListener("click", async (event) => {
5       const radioButton = document.querySelector("#explainWords");
6       if (radioButton && !radioButton.checked) {
7         return;
8       }
9       let leftSideTag = span.closest("p");
10      let rightSideTag = leftSideTag.nextElementSibling;
11      sentence_of_origin = span.closest(".sentence");
12
13      var context = "";
14      for (const child of sentence_of_origin.children) {
15        context = context + " " + child.textContent;
16      }
17      const word = event.target.textContent;
18      const response = await fetch(`http://localhost:5000/look-up-word`, {
19        method: "POST",
20        headers: { "Content-Type": "application/json" },
21        body: JSON.stringify({ word: word, sentence: context }),
22      });
23      result = await response.json();
24
25      if (result.result == "error") {
26        alert("Incorrect API key provided: " + result.word);
27      } else {
28        var pos_tag = result.result.split(' | ')[0];
29        var definition = result.result.split(' | ')[1];
30        let table = document.querySelector(".table-glossary");
31        let newRow = table.insertRow(-1);
32        let cell1 = newRow.insertCell(0);
33        let cell2 = newRow.insertCell(1);
34        let cell3 = newRow.insertCell(2);
35        cell1.innerHTML = result.word;
36        cell2.innerHTML = pos_tag;
37        cell3.innerHTML = definition;
38      }
39    });
40  });

```

41 }) ;

Listing 3.21: Een woord aan de woordenlijst toevoegen in het scholierencomponent.

Ten slotte moet de front end specifieke woorden kunnen markeren. De front-end beschikt al over de PoS-tags. Zo kan de front end met de JS-functie in listing 3.22 deze woorden een specifieke kleur geven als markering. De listing toont enkel het markeren van zelfstandige naamwoorden. Daarnaast kan de front end ook adjektieven uit de tekst verwijderen zonder taalmodel. Zo hoeft de JS-functie enkel de span-tags van de klasse 'adj' verwijderen uit de document.

```
1 const nouns = document.getElementById( 'noun-show' );
2
3 nouns.addEventListener( 'change' , function () {
4     if (this.checked) {
5         const color = document.getElementById( 'colorForNouns' ).value;
6         const elements = document.querySelectorAll( "span.noun" );
7         elements.forEach(function (element) {
8             element.style.color = color;
9         });
10    } else {
11        const elements = document.querySelectorAll( "span.noun" );
12        elements.forEach(function (element) {
13            element.style.color = "black";
14        });
15    }
16});
```

Listing 3.22: Zelfstandige naamwoorden in het scholierencomponent markeren.

3.3.2. De opzet voor een lokale webtoepassing.

Via commandline kunnen eindgebruikers het prototype opstarten. Het prototype online plaatsen valt buiten de scope van dit onderzoek, maar Docker biedt wel een alternatief om een eenduidige opzet van het prototype te garanderen. Omdat het prototype enkel API's van taalmodellen aanspreekt, werkt het prototype met één Docker-container. Listing D.5 toont de gebruikte code van de Dockerfile. Voor het opstarten van de webapplicatie moet de Docker-container eerst de benodigde word-embeddings van Spacy installeren. Een scriptbestand in Powershell, zoals weergegeven in D.6, of Bash zoals weergegeven in D.7, maakt de opstart van deze webapplicatie intuïtiever dan via commandline. Het Pipreq-commando maakt een lijst van python-bibliotheken die Docker vooraf moet installeren. Voor een efficiënte ontwikkeling, moet de Docker-opzet pas laat in het ontwikkelproces gebeuren. Het requirements bestand moet alle nodige Python en front end packages bevatten. Dit proces als eerste laten verlopen kan de versies tegen het einde van de ontwikkeling gedateerd maken. Ten slotte moeten ook alle lettertypen zich in de lokale map bevinden, want Pandoc moet over alle lettertypen beschikken om

een nieuw pdf of docx-bestand te kunnen maken.

3.3.3. Experimenten en vergelijkende studie met het prototype.

Ten slotte komen de laatste twee fasen aan bod. Bij deze twee laatste fasen van deze onderzoeks methode achterhaalt het onderzoek of dit prototype voldoet aan twee zaken. Enerzijds achterhaalt het onderzoek of het prototype voldoet aan de opgestelde functionaliteiten uit het Moscow-schema, weergegeven in figuur 3.4. Het onderzoek toetst daarmee het prototype af en vergelijkt nadien met de andere toepassingen. Vervolgens vergelijkt het onderzoek de aangemaakte wetenschappelijke artikelen, met de oorspronkelijke wetenschappelijke artikelen en referentie-teksten door MTS.

4

Resultaten

In dit hoofdstuk worden de resultaten uit de requirementsanalyse, vergelijkende studie en de ontwikkeling van het prototype besproken.

4.1. Ontbrekende functies bij AI-toepassingen

Op basis van de testen kan de volgende criteria worden afgetoetst, weergegeven in tabel 4.1.

Richtlijn	E1	E2	E3	O1	O2	O3	O4	O5
Rekening houden met doelgroep	-	-	-	-	-	-	P2	P2
Woorden met minder lettergrepen gebruiken	-	X	-	X	-	-	P1-6	P1-6
Extra uitleg schrijven bij zinnen	-	X	-	-	-	-	P1-3	P1-3
Paragrafen herschrijven zodat ze eerst uitleg geven op een high-level niveau	-	-	-	-	-	-	P2	P2
Woordenlijst aanmaken	X	X	X	X	-	-	P6	P6
Synoniemenlijst aanmaken	-	X	-	-	-	-	P6	P6
Idiomen vervangen door eenvoudigere synoniemen	-	-	-	X	-	-	P1-3,6	P1-3,6
Zinnen inkorten	-	-	-	X	X	X	P3-5	P3-5
Verwijswoorden aanpassen	-	-	-	-	-	-	P3	P3
Voorzetseluitdrukkingen aanpassen	-	-	-	-	-	-	P3	P3
Samengestelde werkwoorden aanpassen	-	-	-	-	-	-	P3	P3
Actieve stem toepassen	-	-	-	-	-	-	-	-
Enkel regelmatige werkwoorden gebruiken	-	-	-	-	-	-	P3	P3
Achtergrondkleur aanpassen	X	X	X	-	-	-	-	-
Woord- en karakterspatiëring	-	X	X	-	-	-	-	-
Consistente lay-out	X	X	X	-	-	-	P1-6	P1-6
Duidelijk zichtbare koppenstructuur	-	-	-	-	-	-	-	-
Huidige positie benadrukken	-	-	-	-	-	-	-	-
Waarschuwingen geven omtrent formulieren en sessies	-	-	-	X	-	X	-	-
Inhoud visueel groeperen	-	X	X	-	-	-	-	-
Tekst herschrijven als tabel	-	-	-	-	-	-	P4, P6	P4, P6
Tekst herschrijven als opsomming	-	-	-	-	-	-	P5	P5
Artikel opladen als PDF	X	X	X	X	X	X	-	-
Artikel opladen als <i>plain-text</i>	-	-	-	-	-	-	P1-6	P1-6
Artikel opladen via link	-	-	-	-	X*	-	P1-6*	-

Tabel 4.1: Afgetoetste criteria volgens de experimenten.

Huidige door de overheid erkende software kunnen woorden- en synoniemenlij-

ten genereren na een handmatige selectie van de woorden. Toch kunnen de geteste softwarepakketten geen syntactische vereenvoudiging toepassen op de oorspronkelijke tekst. Daarnaast ontbreken deze erkende software de nodige opmaakopties om een gepersonaliseerde leeservaring aan te bieden. Online beschikbare tools zijn in staat om ATS op zowel Nederlandstalige als Engelstalige wetenschappelijke artikelen toe te passen. Echter houdt geen enkele tool (bewust) rekening met de doelgroep. Daardoor worden alle moeilijke woorden indien mogelijk vervangen door een eenvoudiger synoniem. Indien er geen eenvoudiger synoniem is, dan wordt het woord behouden in de vereenvoudigde tekst en het taalmodel voegt geen ondersteunende definitie toe. Daarnaast maken deze geteste tools geen gebruik van gepersonaliseerde opmaakopties, noch van de webtoepassing noch van het uitvoerbestand. Alle prompts bij ChatGPT en Bing Chat kunnen de benoemde technieken voor lexicale vereenvoudiging toepassen. Enkel schrijven naar de actieve verloopt stroef en alle prompts resulteren in een zin met identieke semantiek, maar geschreven in de passieve stem. De twee chatbots voldoen niet aan de gebruiken een statische weergave en zijn daarmee niet personaliseerbaar. De twee chatbots zijn als enige in staat om structurele aanpassingen in de vorm van opsommingen of tabelstructuren toe te passen op de oorspronkelijke tekst.

ChatGPT en Bing Chat bewijzen dat gepersonaliseerde ATS met evenwaardige kwaliteiten als gepersonaliseerde MTS mogelijk is, maar de online toepassingen ontbreken de nodige opmaakopties om een gepersonaliseerde leeservaring aan te kunnen bieden. Alsook kunnen deze tools geen wetenschappelijke artikelen op een intuïtieve manier inlezen, maar vereisen deze het manueel extraheren van de tekstinhoud om vervolgens deze in de chatbot chunk-by-chunk in te voeren. Daarnaast zijn deze toepassingen niet in staat om teksten naar de actieve stem te schrijven bij geen van de uitgeteste prompts. Passieve zinnen bij de verschillende prompts resulteren in een zin met dezelfde semantiek, maar opnieuw in de passieve vorm. Enkel indien expliciet aangegeven, houdt GPT-3 rekening met de doelgroep waar andere toepassingen niet ertoe in staat zijn.

4.2. Geschikte taalmodel voor gepersonaliseerde tekstvereenvoudiging met ATS

De vergelijkende studie beoordeelt de uitvoer van de uitgeteste taallmodellen, beschreven in 3.6, met een subjectieve en een objectieve benadering. Zo achterhaalt deze onderzoeks methode welk taalmodel of LLM beter aansluit bij het aanbieden van gepersonaliseerde ATS voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

Zowel A1 en A2 bevatten meer zinnen na het vereenvoudigen met T1, T2 en T3 ver-

geleken met de oorspronkelijke tekst, zoals weergegeven in tabel ???. Daarnaast gebruiken T1, T2 en T3 gemiddeld minder woorden per zin dan het oorspronkelijke artikel. Alleen P3 van T4 slaagt erin om gemiddeld minder woorden per zin te gebruiken dan de oorspronkelijke en de referentieteksten van zowel leerlingen en de referentieteksten van leerkrachten, vergeleken met P1 en P2 die elk gemiddeld meer dan 19 woorden per zin gebruiken, zoals te zien is in figuren 4.1 en 4.2.

Bron	Zinnen in A1	Zinnen in A2
Oorspronkelijk	78	159
MTS (door leerkracht)	43	45
MTS (door leerling)	n.v.t.	50
T1	101	209
T2	82	209
T3	100	209
T4 P1	61	98
T4 P2	89	133
T4 P3	39	55

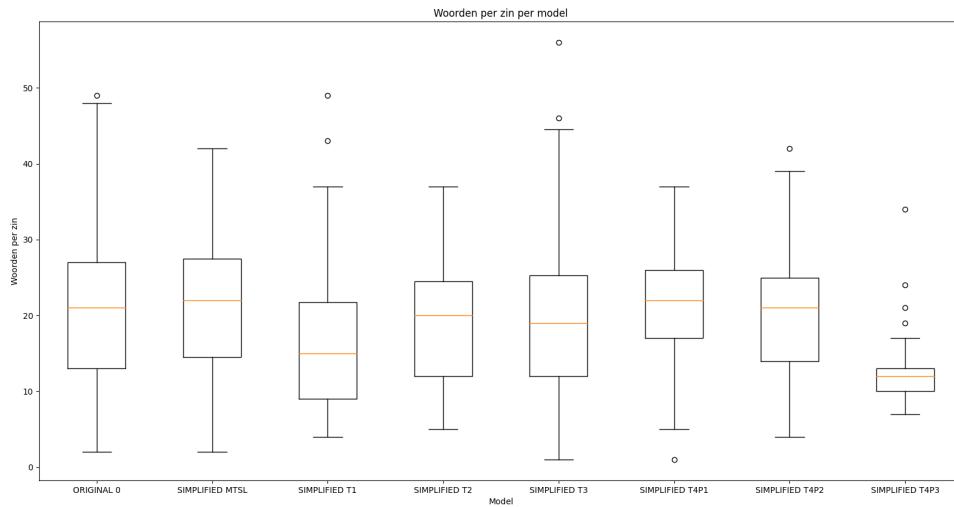
Tabel 4.2: Aantal zinnen (gemeten met Spacy sentence embeddings) per tekst.

De FRE-scores van alle geteste taalmodellen en MTS-referentieteksten zijn niet significant hoger of lager dan die van het oorspronkelijke wetenschappelijke artikel, zoals weergegeven in figuren 4.3 en 4.4. Evenzo zijn de FOG-scores ook niet significant hoger of lager bij de vereenvoudigde wetenschappelijke artikelen, zoals aangegeven in figuren 4.5 en 4.6.

T1, T2 en T3 gebruiken bij zowel A1 als A2 langere woorden vergeleken met de oorspronkelijke tekst, in tegenstelling tot alle T4 prompts die wel langere woorden wegwerken en zo een gelijk resultaat bekomen als de MTS referentieteksten. Deze verhouding wordt aangewezen in figuur 4.9 en figuur 4.10. Daarnaast vervangen T1, T2 en T3 ook minder complexe woorden vergeleken met T4 en de MTS-referentietekst. Dit verschil wordt geïllustreerd in figuur 4.7 en figuur 4.8.

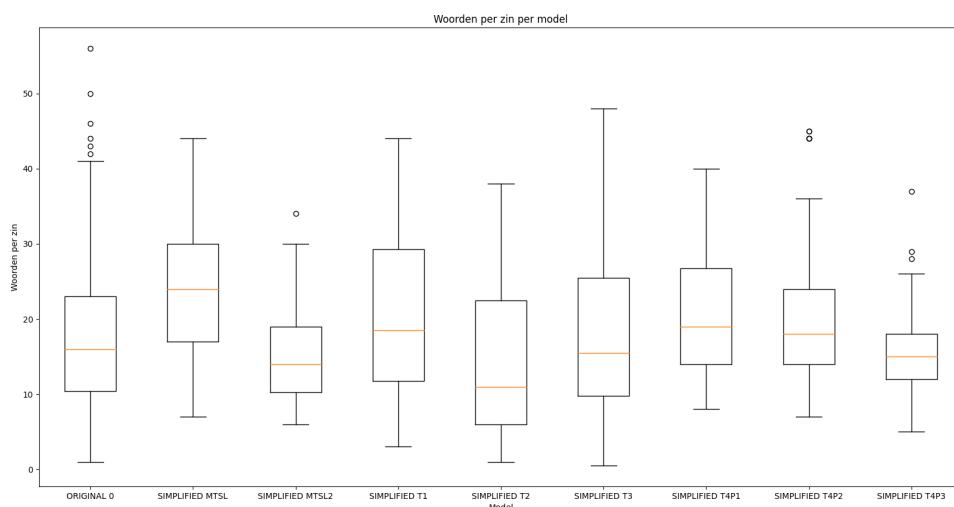
Ten slotte gebruiken T1, T2 en T3 minder hulpwerkwoorden en vervoegingen van het werkwoord 'zijn'. Daarnaast tonen 4.11 en 4.12 aan dat alle prompts van T4 een gelijke trend volgen met het aantal vervoegingen en hulpwerkwoorden vergeleken met de MTS-referentieteksten.

- Acroniemen behouden
- Inschatting van de doelgroep



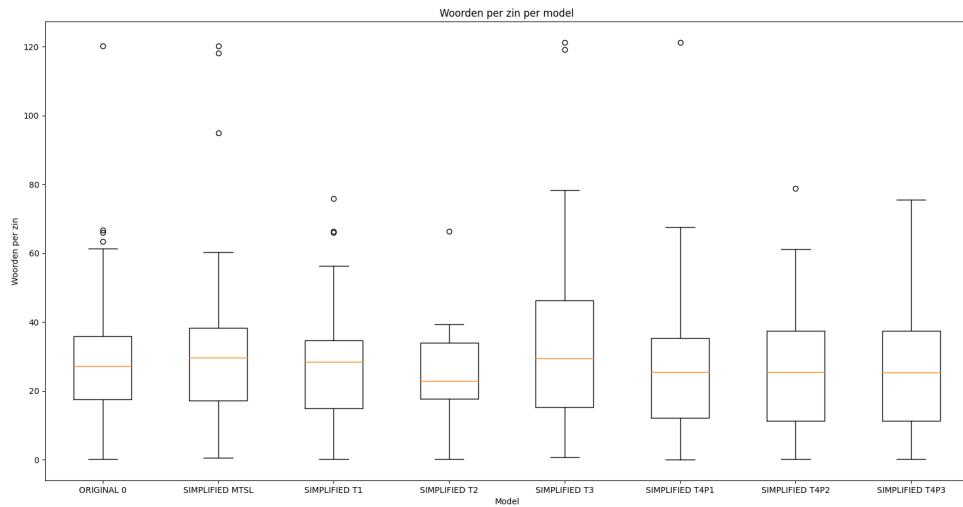
Figuur (4.1)

Overzicht van het minimum, maximum en gemiddeld aantal woorden per zin per model in A1.

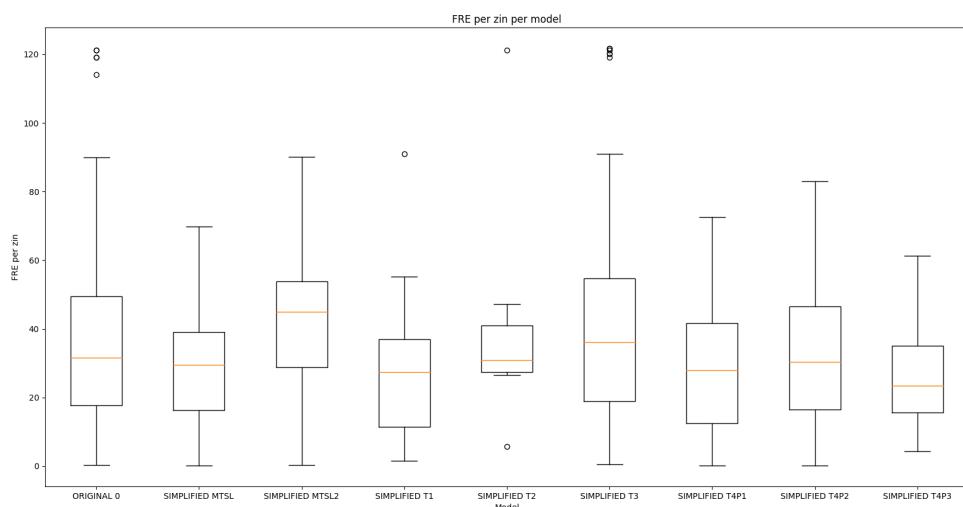


Figuur (4.2)

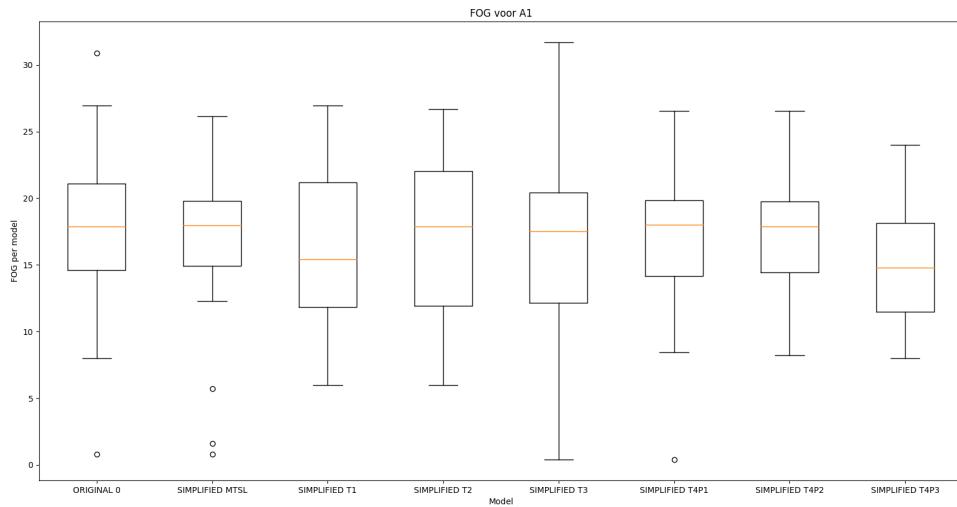
Overzicht van het minimum, maximum en gemiddeld aantal woorden per zin per model in A2.

**Figuur (4.3)**

Boxplot van de FRE-scores voor A1.

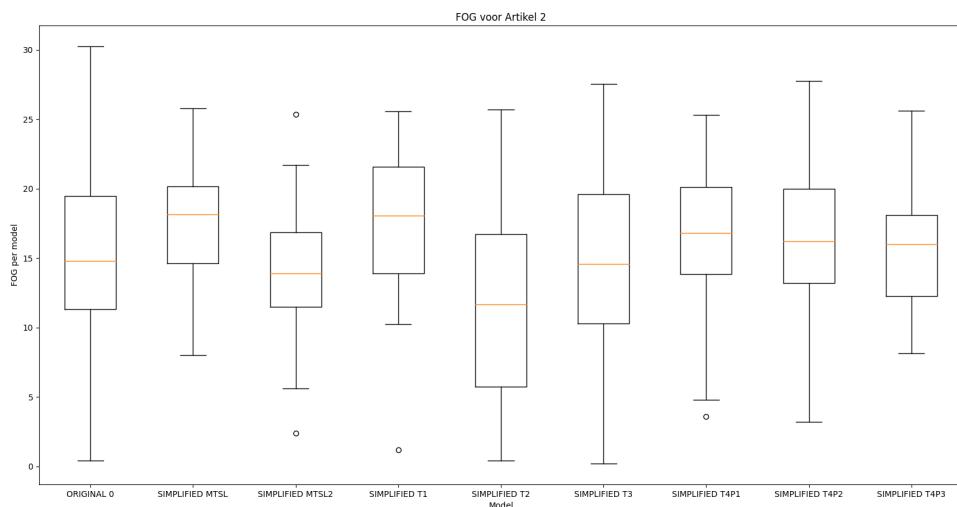
**Figuur (4.4)**

Boxplot van de FRE-scores voor A2.



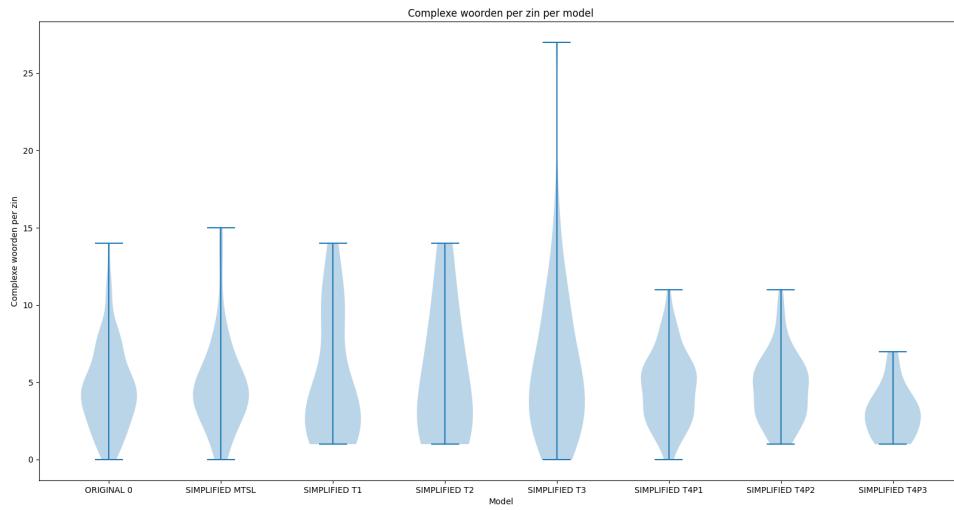
Figuur (4.5)

Boxplot van de FOG-scores voor A1.

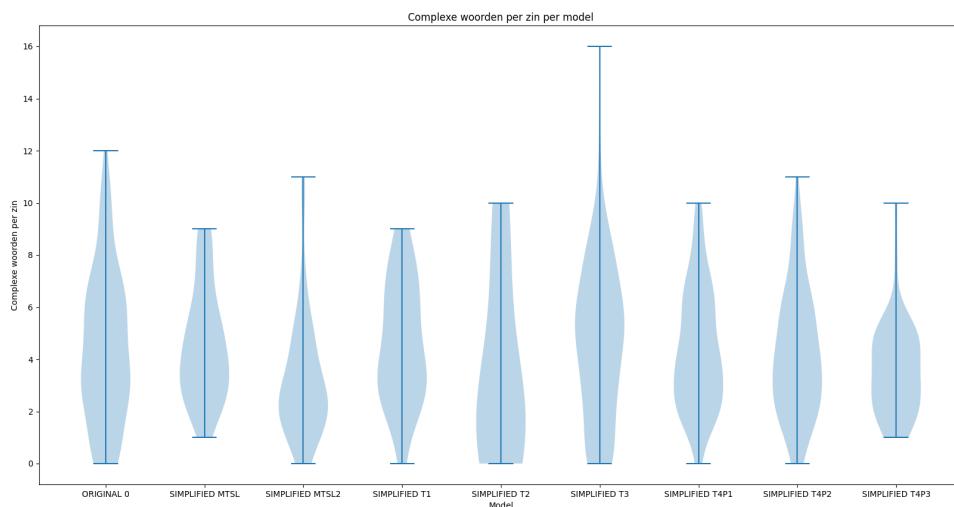


Figuur (4.6)

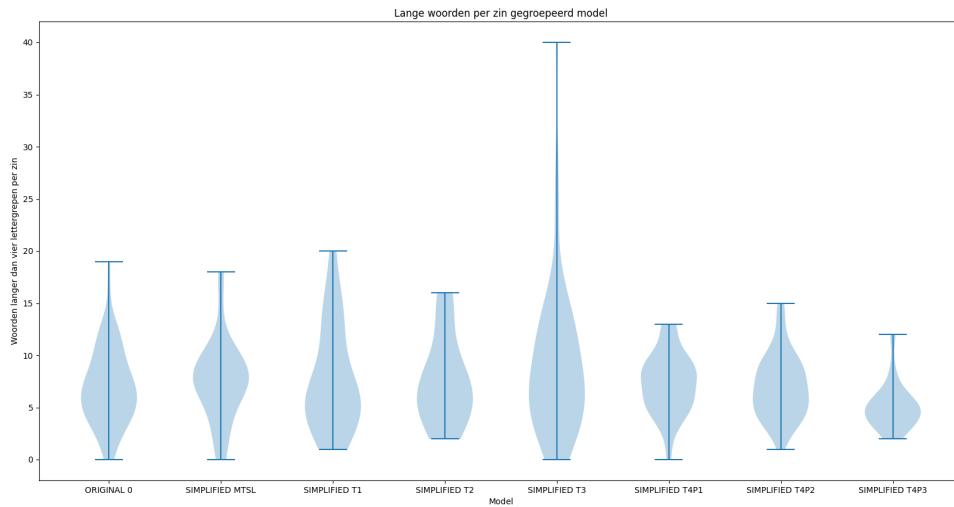
Boxplot van de FOG-scores voor A2.

**Figuur (4.7)**

Gemiddeld aantal complexe woorden per zin gegroepeerd op model voor A1.

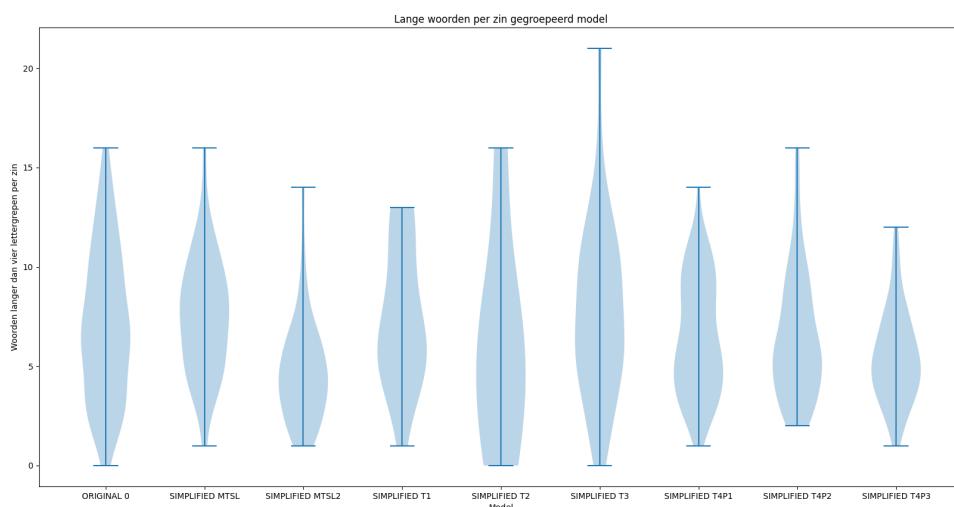
**Figuur (4.8)**

Gemiddeld aantal complexe woorden per zin gegroepeerd op model voor A2.



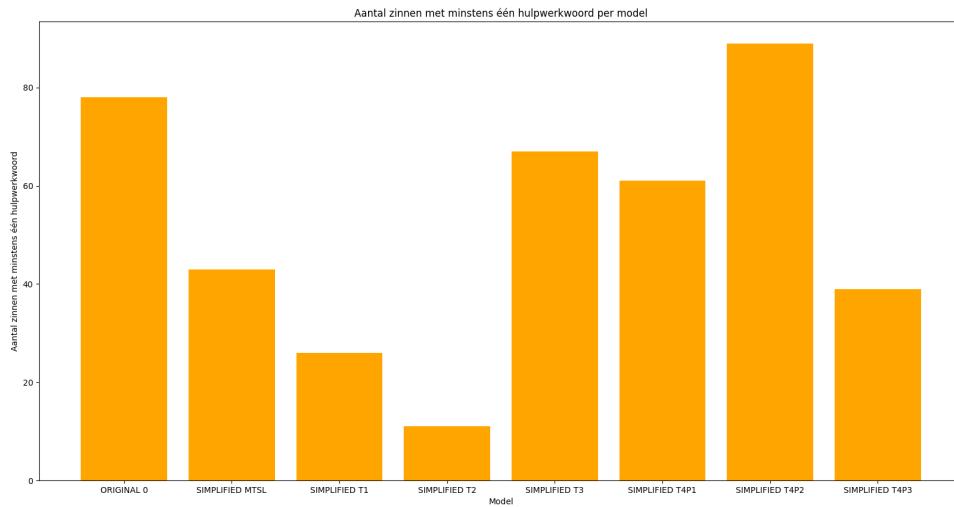
Figuur (4.9)

Gemiddeld aantal lange woorden per zin gegroepeerd op model voor A1.

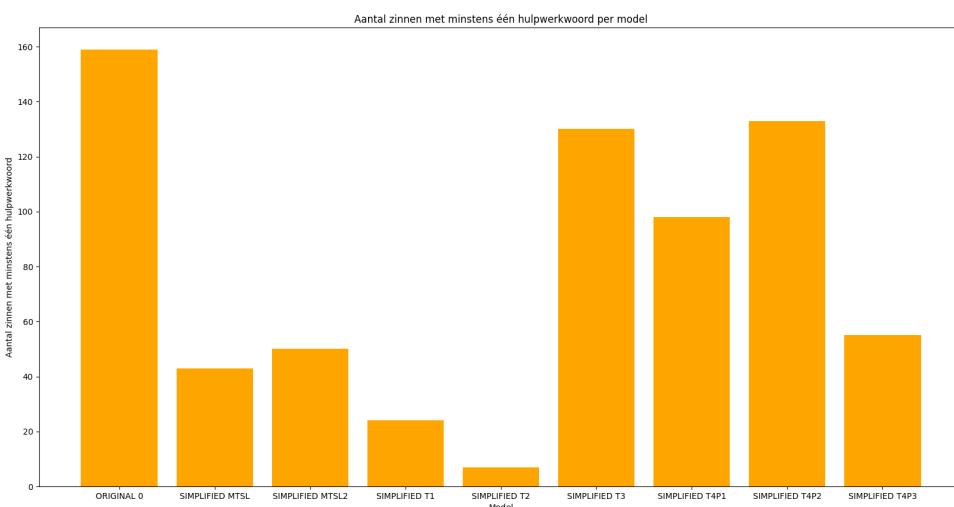


Figuur (4.10)

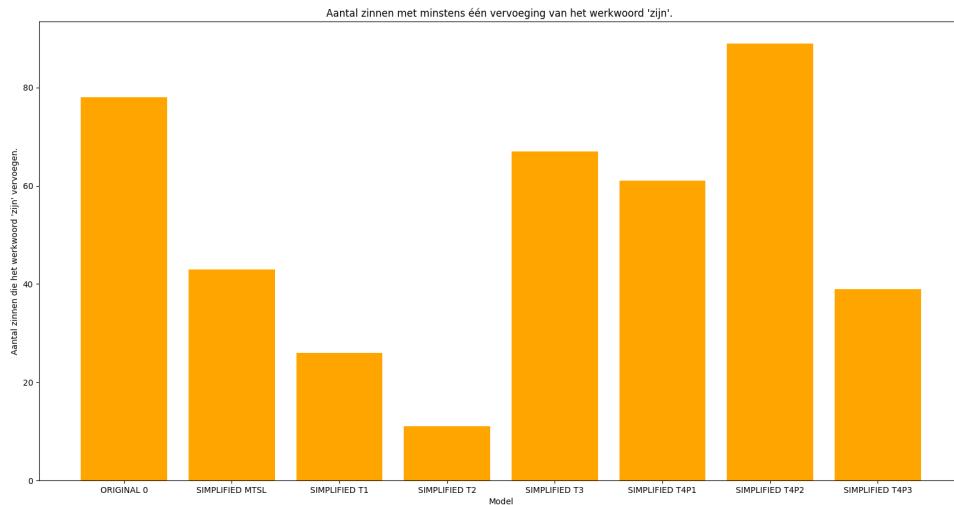
Gemiddeld aantal lange woorden per zin gegroepeerd op model voor A2.

**Figuur (4.11)**

Gemiddeld aantal hulpwerkwoorden per zin gegroepeerd op model voor A1.

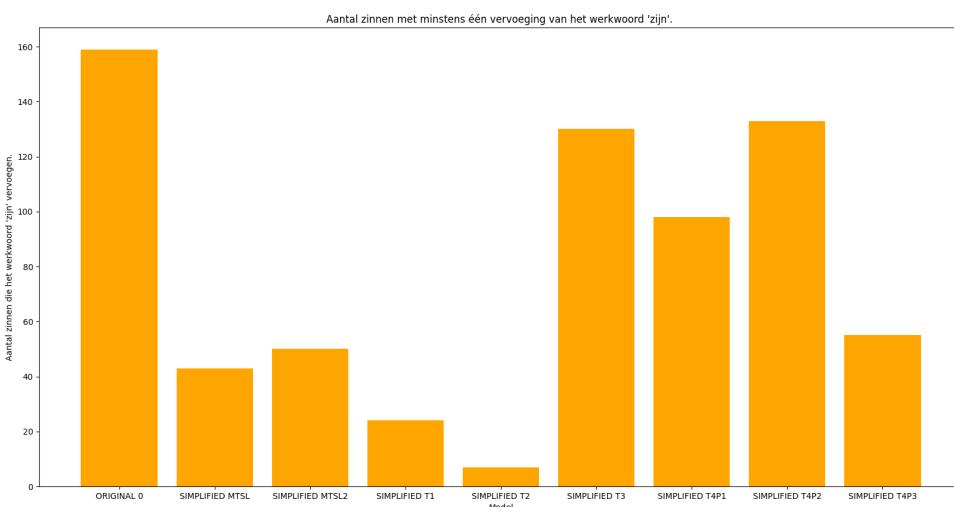
**Figuur (4.12)**

Gemiddeld aantal hulpwerkwoorden per zin gegroepeerd op model voor A2.



Figuur (4.13)

Gemiddeld aantal vervoegingen van het werkwoord 'zijn' per zin gegroepeerd op model voor A1.



Figuur (4.14)

Gemiddeld aantal vervoegingen van het werkwoord 'zijn' per zin gegroepeerd op model voor A2.

- Behoud van kern- en bijzaken
- Bronvermelding behouden
- Formaatwijzigingen
- Citeren en parafraseren

T4P1 en T4P2 vertalen Engelstalige vaktermen naar het Nederlands. Zo blijft de afkorting voor DPKIA intact, maar vertaalt T4P1 hetzelfde woord naar het Nederlands. T1, T2, T3 en T4P3 houden hier echter geen rekening mee en behouden de oorspronkelijke versie van de tekst. Bij de handmatige vereenvoudiging schrijft de auteur alle afkortingen voluit, zoals beschreven in de richtlijnen.

Alle taalmodellen kunnen lexicale vereenvoudiging toepassen. De handmatig vereenvoudigde referentieteksten bevatten zinnen die vakjargon gebruiken op het niveau van 15 tot 18 jarige studenten. T4P1 kan uitleg tussen ronde haakjes schrijven, wanneer het geen eenvoudiger synoniem kan vinden. T4P1, T1, T2 en T3 passen woorden aan, maar schrijven geen extra uitleg. T4P3 past deze techniek minder toe dan de vooraf vermelde taalmodellen.

Geen taalmodel wijkt af van de hoofdgedachte van het oorspronkelijke wetenschappelijk artikel. Hoewel T1, T2 en T3 deels afgebroken zinnen kan genereren, bevatten deze zinnen de hoofdgedachte. Tenslotte verwerken T1, T2 en T3 de APA- en California bronvermeldingen niet in de vereenvoudigde teksten. Hoewel T4 deze wel verwerkt, bevat de tekst na een vereenvoudiging deze bronvermeldingen niet meer.

T4P3 maakt de zinnen korter door langere zinnen op te splitsen. T1, T2 en T3 behalen een gelijke zinslengte als dat van de oorspronkelijke zin. T4P1 en T4P2 voegen

4.3. Het prototype voor tekstvereenvoudiging met ATS vergeleken met top-of-the-line tools.

De ontwikkeling van een prototype voor een gepersonaliseerde ATS van wetenschappelijke artikelen heeft tot doel om ontwikkelaars inzicht te verschaffen in de mogelijkheden om een coherente en lokale webtoepassing te ontwikkelen. Na evaluatie blijkt het prototype te voldoen aan de gespecificeerde functionaliteiten zoals vastgesteld in het Moscow-schema of tabel 3.4.

Hiermee overtreft het prototype alle andere geteste tools in tabel 2.9 op alle fronten. Eerdere belemmeringen, zoals het wijzigen van het formaat, waren enkel mogelijk via de commandline of via een chatbot. Het huidige prototype heeft echter intuïtieve handelingen geïmplementeerd om deze obstakels te overwinnen.

Prototype voor tekstvereenvoudiging**Figuur (4.15)**

Een mogelijke weergaven van de homepagina.

Het prototype is uitgerust met functionaliteiten waarmee wetenschappelijke artikelen op consistente wijze kunnen worden ingeladen, waarna de gebruikers gepersonaliseerde ATS kunnen toepassen. Deze functionaliteiten zijn vergelijkbaar met andere geteste tools in de requirementsanalyse, maar gaan verder dan wat momenteel beschikbaar is in chatbots die geavanceerde logica gebruiken voor gepersonaliseerde ATS. Zowel scholieren als docenten kunnen ook de opmaakopties van de toepassing aanpassen aan hun persoonlijke voorkeuren bij het lezen van wetenschappelijke artikelen.

The screenshot shows a web-based application interface. At the top, there's a header bar with a search bar containing 'Geef de titel hier in' and 'Zoek het onderwerp van de'. Below the header, there's a main content area divided into several sections:

- Inleiding**: A text block from 'Vrije Universiteit Brussel De controle op het gebruik van algorithmische surveillance- onder druk ? Een exploratie door de lens van relationele ethiek' by Van Brakel, Rosanne. It discusses the use of big data and AI in surveillance.
- Tools voor de leerkracht**: A panel with several checkboxes for text styling:
 - Werkwoorden tonen
 - Zelfstandige naamwoorden tonen
 - Adjektieven tonen
 - Za verwijderen
 - Woord toevoegen aan woordenschat
 Sub-options include 'Lettertype voor titel' set to Arial, and checkboxes for 'Aan' (checked) and 'Andere' (unchecked), with 'Regelende' selected. There are also input fields for 'Woord-spativering (in cm)' (0.5) and 'Marge (in cm) van document' (1).
- Samenvatting op maat**: A panel with checkboxes for summarization:
 - Moeilijke woordenschat vervangen
 - Cijferwaarden interpreteren
 - Tenseconstructies aanpassen
 - Vervuwoorden aanpassen
 - Voorzetbeschrijvingen vervangen
 - Samengestelde werkwoorden vervangen
 - Bronvermelding behouden
 - Schrijven als opsomming
- Gebruikte API-sleutels**: A panel stating 'Geen HuggingFace sleutel werd opgegeven.' and 'Geen GPT-3 sleutel werd opgegeven.'
- Woordenlijst**: A panel with a link 'Volledig document samenvatten'.

Figuur (4.17)

Een mogelijke weergave van het lerarencomponent met het wetenschappelijk artikel van Van Brakel (2022) als input.

This screenshot shows the 'Instellingen' (Settings) page of the prototype. It includes the following sections:

- Achtergrondkleur**: A color selection box.
- Tekstenmerken**: A panel with controls for lettering:
 - Lettertype: Arial (selected)
 - Woord Spativering: 0.5
 - Regelhoogte: 1
 - Lettergrootte: 21
 - Alignering: Links (selected)
 - Inhalingen aanpassen
- API-sleutels**: A panel with buttons for 'GPT API KEY' and 'HuggingFace API KEY'.

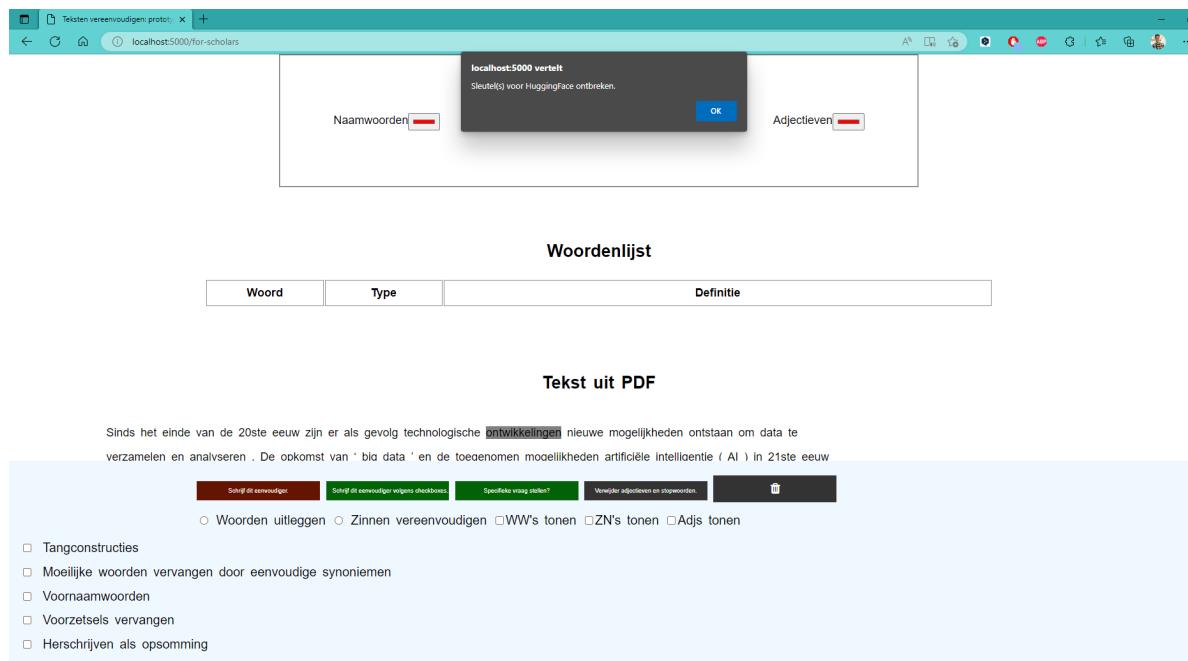
Figuur (4.16)

Voorbeeldweergave van de instellingenpagina.

Bovendien stelt het prototype gebruikers in staat om op basis van gegeven parameters automatisch personaliseerbare PDF- en Word-documenten te genereren.

4.3. Het prototype voor tekstvereenvoudiging met ATS vergeleken met top-of-the-line tools.

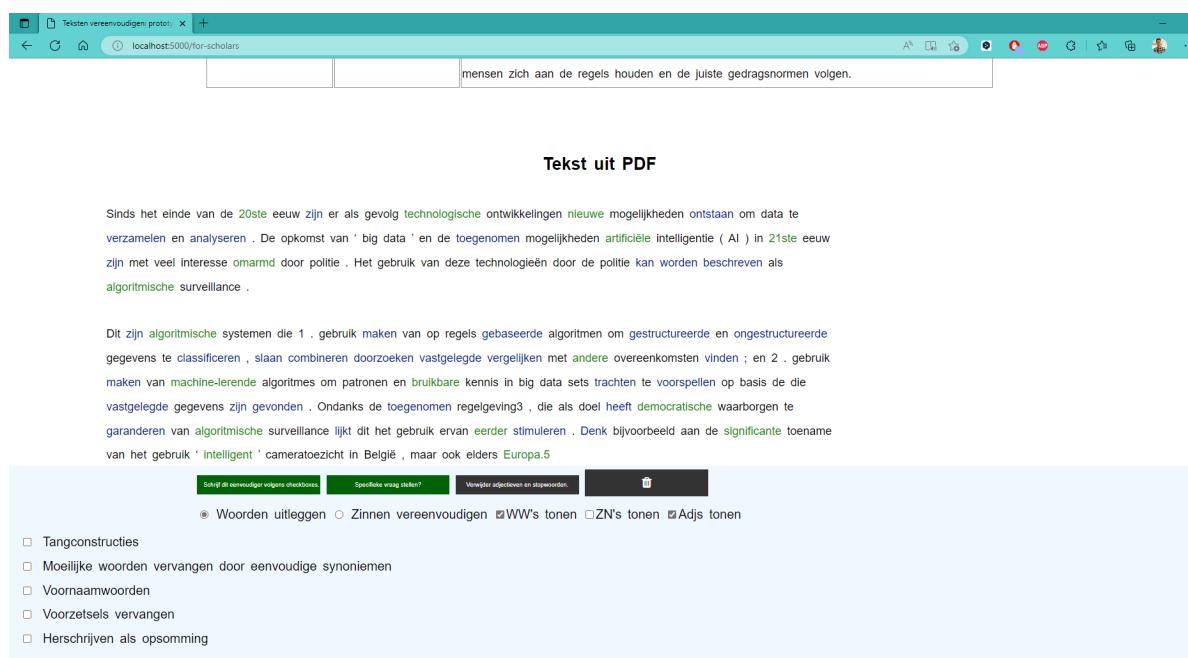
89



Figuur (4.18)

Een voorbeeldweergave van het scholierencomponent.

Figuur 4.19 toont een voorbeeldweergave van deze functionaliteit.



Figuur (4.19)

Een voorbeeldweergave van de toepassing van PoS-tagging bij het scholierencomponent.

The screenshot shows a browser window with the URL localhost:5000/for-scholars. The title bar says "Teksten vereenvoudigen: proto...". The main content area is titled "Tekst uit PDF". Below it is a text editor containing a paragraph about algorithmic surveillance. At the bottom of the editor are several buttons: "Schrijf dit evenredig vóórgelegde checkblokken", "Specifieke vraag stellen?", "Verwijder allechecken en stopcondities", and a trash bin icon. Below these buttons is a radio button group with the text "Woorden uitleggen" followed by three checkboxes: "Zinnen vereenvoudigen", "WW's tonen", "ZN's tonen", and "Adjs tonen". There is also a checked checkbox for "Tangconstructies". To the right of the text editor is a list of other options: "Moeilijke woorden vervangen door eenvoudige synoniemen", "Voornaamwoorden", "Voorzetsels vervangen", and "Herschrijven als opsomming". The text in the editor is summarized as follows:

"In Simplify the sentences in the given text and appositions, summary\n :return: A list of simplified sentences divided by a ']' sign\n //n Dit zijn algoritmische systemen die 1 . gebruik maken van op regels gebaseerde algoritmen om gestructureerde en ongestructureerde gegevens te classificeren , slaan combineren doorzoeken vastgelegde vergelijken met andere overeenkomsten vinden ; en 2 . gebruik maken van machine-lerende algoritmes om patronen en bruikbare kennis in big data sets trachten te voorspellen op basis de die vastgelegde gegevens zijn gevonden . Ondanks de toegenomen regelgeving3 , die als doel heeft democratische waarborgen te garanderen van algoritmische surveillance lijkt dit het gebruik ervan eerder stimuleren . Denk bijvoorbeeld aan de significante toename van het gebruik ' intelligent ' cameratoezicht in België maar ook elders Europa.5in "

Figuur (4.20)

Stap 1 van een gepersonaliseerde tekstvereenvoudiging in het scholierencomponent.

The screenshot shows a browser window with the URL localhost:5000/for-scholars. The title bar says "Teksten vereenvoudigen: proto...". The main content area is titled "Tekst uit PDF". Below it is a text editor containing the same paragraph as in Figure 4.20. At the bottom of the editor is a list of bullet points:

- Algoritmische systemen gebruiken regels-gebaseerde algoritmen om gegevens te classificeren, combineren, doorzoeken, vergelijken en overeenkomsten te vinden
- Machine-lerende algoritmen trachten patronen en bruikbare kennis in big data sets te voorspellen
- Hoewel er regelgeving is om democratische waarborgen te garanderen, lijkt dit het gebruik van algoritmische surveillance te stimuleren
- Er is een significante toename van het gebruik van 'intelligent' cameratoezicht in België en andere delen van Europa

Figuur (4.21)

Stap 3 van een gepersonaliseerde tekstvereenvoudiging in het scholierencomponent.

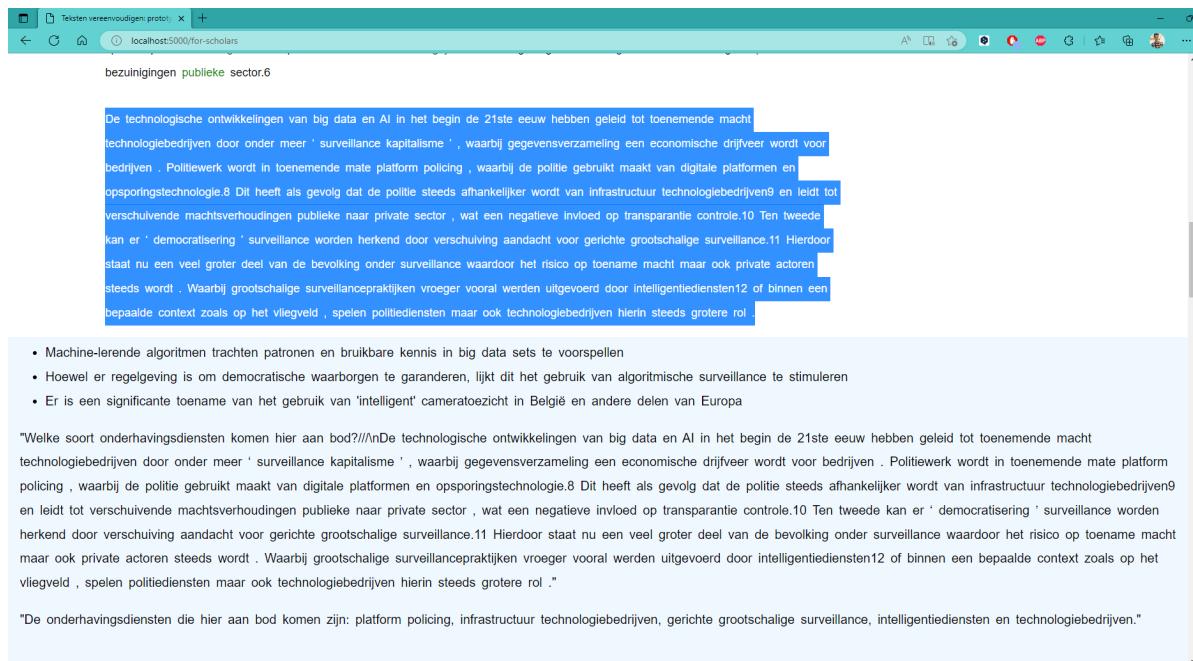
4.3. Het prototype voor tekstvereenvoudiging met ATS vergeleken met top-of-the-line tools.

91



Figuur (4.22)

Stap 1 bij het stellen van een specifieke vraag bij gemaakte teksten.



Figuur (4.23)

Stap 2 bij het stellen van een specifieke vraag bij gemaakte teksten.



5

Conclusie

Deze scriptie tracht een antwoord te bieden op de volgende onderzoeksraag:

- Hoe kan een wetenschappelijk artikel automatisch vereenvoudigd worden, gericht op de unieke noden van scholieren met dyslexie in de derde graad middelbaar onderwijs?

De requirementsanalyse bood nieuwe inzichten in de huidige toepassingen van ATS. Zo beschikken online tools te weinig over gepersonaliseerde ATS-functionaliteiten, zoals gebleken in sectie 3.1. Daarnaast maken weinig tools gebruik van gepersonaliseerde weergaveopties, die echter een bewezen effect hebben op het leesbegrip van een scholier met dyslexie in de derde graad van het middelbaar onderwijs.

Uit de vergelijkende studie blijkt dat GPT-3 de leesbaarheid van een wetenschappelijke tekst kan bevorderen door gebruik te maken van eenvoudigere synoniemen en structurele aanpassingen, waaronder het schrijven van tekst als een opsomming. Andere geteste taalmodellen uit HuggingFace behalen zwakkere resultaten en vereisen een extra vertaalfase, die redundant is bij het aanspreken van de GPT-3 API.

Uit de ontwikkeling van het prototype voor gepersonaliseerde ATS bleek dat *open-source* AI en NLP-technologieën voldoende hoogstaand genoeg zijn om kwaliteitsvolle tekstvereenvoudigingssoftware te ontwikkelen. Zo kunnen ontwikkelaars gebruikmaken van PDFMiner om teksthoud uit wetenschappelijke artikelen te extraheren, van OpenAI's GPT-3 model via de API om gepersonaliseerde ATS mogelijk te maken en ten slotte van Pandoc om dynamische en gepersonaliseerde PDF-documenten automatisch te genereren. Binnen een webapplicatie kunnen een-duidige handelingen, gebouwd in JavaScript en HTML&CSS, complexe command-linehandelingen afhandelen.

Ontwikkelaars hebben toegang tot T1, T2 en T3 via HuggingFace voor lexicale vereenvoudigingstaken. Deze taalmodellen zijn echter ontoereikend voor gepersonaliseerde tekstvereenvoudiging en daarom is T4 een geschikter model voor het vereenvoudigen van wetenschappelijke artikelen op maat. GPT-3 presteert goed op gepersonaliseerde vereenvoudigingstaken, maar het is belangrijk om op te merken dat geen enkel taalmodel de doelgroep altijd nauwkeurig kan inschatten. Extra trainingsdata, zoals leerstof of wetenschappelijke artikelen die wel op het niveau van een 16-18-jarige zijn geschreven, kan het model steunen bij de doelgroepsinschatting. Het gebruik van Engelstalige prompts met expliciete vermelding van de gewenste uitvoertaal, resulteert in coerentere teksten dan bij een Nederlands-talige prompt.

6

Discussie

Dit onderzoek gebruikt drie onderzoeksmethoden om te bepalen hoe ontwikkelaars een optimale vorm van gepersonaliseerde ATS kunnen bieden aan scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

Uit de resultaten van de requirementsanalyse blijkt dat zowel erkende toepassingen als online tools onvoldoende functionaliteiten bieden voor gepersonaliseerde ATS en te weinig opties bieden voor personalisering van opmaak. Dit resultaat komt overeen met de verwachting dat bestaande tools niet specifiek gericht zijn op gepersonaliseerde ATS voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Mogelijke verklaringen hiervoor zijn de populariteit van samenvattingstools in vergelijking met vereenvoudigingstools, de complexiteit die gepaard gaat met de ontwikkeling van dergelijke toepassing en het gebrek aan initiatief binnen dit vakgebied.

Hoewel ChatGPT en Bing Chatbot functionaliteiten bieden voor gepersonaliseerde ATS, ontbreken eenduidige handelingen waardoor gebruikers moeite kunnen hebben met het vereenvoudigen van wetenschappelijke artikelen. Daarnaast houdt het model van ChatGPT geen rekening met referenties buiten de getrainde data, wat problemen kan veroorzaken op het gebied van data-integriteit. Hiertegenover staat Bing AI, dat wel rekening houdt met externe referenties en daarmee een goede basis vormt voor ontwikkelaars om meer referentiemateriaal aan te bieden in ondersteunende onderwijsstoepassingen. Verder onderzoek naar de toepassing van deze AI via een mogelijke API is absoluut noodzakelijk en kan baanbrekend zijn voor de onderwijssector. Aan de andere kant is er de mogelijkheid om bestaande toepassingen zoals Kurzweil uit te breiden met functionaliteiten die gepersonaliseerde ATS aanbieden aan scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

Het experiment in de vergelijkende studie maakte gebruik van het GPT-3-model met de Da Vinci 3 engine en is alleen gefinetuned op basis van API-parameters. Zo bevat het ook geen extra vooraf getrainde data van wetenschappelijke artikelen. Uit een vergelijkende studie van taalmodellen voor gepersonaliseerde ATS bleek dat zowel de drie geteste HuggingFace-modellen via API als het geteste GPT-3-model via API in staat waren om moeilijke woorden te identificeren en te vervangen. Hoewel de vrij beschikbare modellen op HuggingFace lexcale vereenvoudiging mogelijk kunnen maken, staan ze in de schaduw van GPT-3, dat als API vrij beschikbaar is voor ontwikkelaars. Het GPT-3-model kan een baanbrekende oplossing bieden voor gepersonaliseerde ATS van wetenschappelijke artikelen, omdat het snel en efficiënt moeilijke woorden kan herkennen in doorlopende tekst en structurele aanpassingen kan maken aan de oorspronkelijke tekst.

Dit resultaat bevestigt de verwachting dat GPT-3 beter in staat is om gepersonaliseerde ATS aan te bieden in vergelijking met vrij beschikbare HuggingFace-taalmodellen. Een mogelijke verklaring hiervoor is de trainingsdata en de complexiteit van het taalmodel, mede door het aantal parameters dat ze bevatten. Er zijn lichte verschillen in lexcale complexiteit tussen de HuggingFace-taalmodellen die getraind waren op wetenschappelijke artikelen en taalmodellen die getraind waren op algemene data. Meer onderzoek is echter nodig om deze verschillen beter te begrijpen binnen de context van wetenschappelijke artikelen. De studie kon geen gebruikmaken van het opvolgende GPT-model, namelijk GPT-4, en hield geen rekening met het hoofdstuk waarin die een zin beoordeelt. Daarom bleven vragen over het verschil na ATS per hoofdstuk in een wetenschappelijk artikel onbeantwoord. Dit vereist verder onderzoek.

LLM's, waaronder GPT-3, kunnen vragen beantwoorden en tekstvereenvoudiging vergemakkelijken. Er is meer onderzoek nodig naar de verschillen op taalgebied in relatie tot de toename van parameters bij grotere taalmodellen, zoals GPT-4 en LLaMa. Verder onderzoek naar de effecten van het meegeven van doelgroepen via prompts is ook nodig. Daarnaast zou toekomstig onderzoek zich kunnen richten op het potentieel van de combinatie van GPT-3 en *full-text-search*-technologieën. Meer onderzoek is nodig naar de verschillen tussen taalmodellen die getraind zijn op wetenschappelijke artikelen en taalmodellen die getraind zijn op algemene data binnen de context van wetenschappelijke artikelen. Er is ook weinig onderzoek gedaan naar het verschil tussen het laten schrijven van prompts en het gebruik van vooraf gedefinieerde prompts, wat verdere aandacht verdient. Er is behoefte aan onderzoek naar het gebruik van nieuwe modellen zoals GPT-4 en Bing AI in het onderwijs. De verschillen tussen FRE en FOG waren onderling miniem. Verder onderzoek is nodig om de bruikbaarheid van deze leesgraadscores te bepalen en te begrijpen hoe ze zich verhouden tot de kwaliteit van de vereenvoudigde tekst.

aan Vlaamse Hogescholen, lieten toe om het prototype voor de webtool te ontwikkelen. Dit prototype dient slechts als een haalbaarheidsmeting voor ontwikkelaars bij het ontwikkelen van dergelijke toepassing. Het is belangrijk dat de lezer zich bewust is van het feit dat de webtool zich baseert op eerder onderzochte kenmerken en technieken die de impact van tekstvereenvoudiging met MTS hebben aangeïntoond bij scholieren met dyslexie.

Daarnaast gebeurde de ontwikkeling van het prototype met het oog op een snelle en eenduidige implementatie van technieken die voordien enkel beschikbaar waren via commandline. Tijdens de ontwikkeling van het prototype is gebleken dat ontwikkelaars met vrij beschikbare middelen en API's in staat zijn om gepersonaliseerde ATS-toepassingen te bieden aan scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Zo kunnen ontwikkelaars het stappenplan volgen om een vergelijkbaar resultaat te behalen, alsook de taken binnenin een projectteam parallel te laten uitvoeren volgens de flowchart.

Dit resultaat komt overeen met de verwachting dat ontwikkelaars over de benodigde tools beschikken om een dergelijk prototype voor gepersonaliseerde ATS te maken. Een mogelijke verklaring hiervoor is de beschikbaarheid van *open-source* tools en Python-bibliotheken die ontwikkelaars in staat stellen complexe taken eenvoudig uit te voeren. Toch moet de lezer zich ervan bewust zijn dat het prototype niet getest is bij het doelpubliek tijdens het begrijpend lezen van een wetenschappelijk artikel. Daarom kan het alleen dienen als een meting van de haalbaarheid voor ontwikkelaars. Er is momenteel een gebrek aan wetenschappelijke vakliteratuur met betrekking tot tekstvereenvoudiging met ATS voor deze specifieke doelgroep.

Logopedisten of studenten in deze richting kunnen dit prototype gebruiken om onderzoek uit te voeren naar het effect op leesbegrip bij scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Onderzoekers binnen het vakdomein secundair onderwijs kunnen de effecten van deze tool observeren bij leerlingen en leerkrachten in het middelbaar onderwijs. Er is echter meer onderzoek nodig om de inzet van gepersonaliseerde ATS-toepassingen en browserextensies voor tekstvereenvoudiging in het onderwijs te verbeteren. Er is behoefte aan een toepassing die alle functionaliteiten kan combineren. Bovendien is er behoefte aan meer onderzoek naar tekstvereenvoudiging met ATS voor de specifieke doelgroep van scholieren met dyslexie. Ten slotte is er onderzoek nodig naar de verbetering van de inzet van gepersonaliseerde ATS-toepassingen en browserextensies voor tekstvereenvoudiging in het onderwijs. Ook moet er onderzoek worden gedaan naar het gebruik van gepersonaliseerde ATS-toepassingen en browserextensies voor tekstvereenvoudiging bij scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

A

Onderzoeksvoorstel

Samenvatting

Ingewikkelde woordenschat en zinsbouw hinderen scholieren met dyslexie in de derde graad van het middelbaar onderwijs bij het begrijpend lezen van wetenschappelijke artikelen. Gepersonaliseerde *automated text simplification* (ATS) helpt deze scholieren bij hun leesbegrip. Daarnaast kan artificiële intelligentie (AI) dit proces automatiseren om de werkdruk bij leraren en scholieren te verminderen. Dit onderzoek achterhaalt met welke technologische en logopedische aspecten AI-ontwikkelaars rekening moeten houden bij de ontwikkeling van een AI-toepassing voor geautomatiseerde en gepersonaliseerde tekstvereenvoudiging. Hiervoor is de volgende onderzoeksraag opgesteld: "Hoe kan een wetenschappelijk artikel automatisch worden vereenvoudigd, gericht op de unieke noden van scholieren met dyslexie in het derde graad middelbaar onderwijs?". Een requirementsanalyse achterhaalt de benodigde functionaliteiten om gepersonaliseerde en geautomatiseerde tekstvereenvoudiging mogelijk te maken. Vervolgens wijst de vergelijkende studie uit welk taalmodel ontwikkelaars kunnen inzetten om de taak van gepersonaliseerde en geautomatiseerde tekstvereenvoudiging mogelijk te maken. De requirementsanalyse wijst uit dat toepassingen om wetenschappelijke artikelen te vereenvoudigen, gemaakt zijn voor een centrale doelgroep en geen rekening houden met de unieke noden van een scholier met dyslexie in het derde graad middelbaar onderwijs. Toepassingen voor gepersonaliseerde ATS zijn mogelijk, maar ontwikkelaars moeten meer inzetten op de unieke noden van deze scholieren.

A.1. Introductie

Het Vlaams middelbaar onderwijs staat op barsten. Werkdruk en stress overspoelen leraren en scholieren. Bovendien is de derde graad van het middelbaar onderwijs een belangrijke mijlpaal voor de verdere loopbaan van scholieren, al hebben

zij volgens Dapaah en Maenhout (2022) dan moeite om grip te krijgen op de vakliteratuur bij STEM-vakken. De STEM-agenda¹ van de Vlaamse overheid moet het STEM-onderwijs tegen 2030 aantrekkelijker te maken, door de ondersteuning voor zowel leerkrachten als scholieren te verbeteren. Toch neemt deze agenda de aanpak van steeds complexere wetenschappelijke taal, zoals beschreven in Barnett en Doubleday (2020), niet op. Wetenschappelijke artikelen vereenvoudigen, op maat van de noden van een scholier met dyslexie in het middelbaar onderwijs, is tijds- en energie-intensief voor leerkrachten en scholieren. Automatische en adaptieve tekstvereenvoudiging biedt hier een baanbrekende oplossing om de werkdruk in het middelbaar onderwijs te verminderen.

Het doel van dit onderzoek is om te achterhalen met welke technologische en logopedische aspecten AI-ontwikkelaars rekening moeten houden bij de ontwikkeling van een adaptieve AI-toepassing voor geautomatiseerde tekstvereenvoudiging. De volgende onderzoeksfrage is opgesteld: "Hoe kan een wetenschappelijk artikel automatisch vereenvoudigd worden, gericht op de verschillende behoeften van scholieren met dyslexie in de derde graad middelbaar onderwijs?". Een antwoord op volgende deelvragen kan de onderzoeksfrage vereenvoudigen. Eerst geeft de literatuurstudie een antwoord op de eerste vier deelvragen. Daarna vormt het veldonderzoek een antwoord op de vijfde deelvraag. Ten slotte beantwoordt de vergelijkende studie de zesde en laatste deelvraag. De resultaten van dit onderzoek zetten AI-ontwikkelaars aan om een toepassing te maken om scholieren met dyslexie te kunnen ondersteunen in de derde graad middelbaar onderwijs.

1. Welke aanpakken zijn er voor geautomatiseerde tekstvereenvoudiging? Aansluitende vraag: "Hoe worden teksten handmatig vereenvoudigd voor scholieren met dyslexie?"
2. Welke specifieke noden hebben scholieren van de derde graad middelbaar onderwijs bij het begrijpen van complexere teksten?
3. Wat zijn de specifieke kenmerken van wetenschappelijke artikelen?
4. Met welke valkuilen bij taalverwerking met AI moeten ontwikkelaars rekening houden?
5. Welke toepassingen, tools en modellen zijn er beschikbaar om Nederlandstalige geautomatiseerde tekstvereenvoudiging met AI mogelijk te maken?
6. Welke functies ontbreken AI-toepassingen om geautomatiseerde én adaptieve tekstvereenvoudiging mogelijk te maken voor scholieren met dyslexie in de derde graad middelbaar onderwijs? Aansluitende vraag: "Welke manuele methoden voor tekstvereenvoudiging komen niet in deze tools voor?"

¹<https://www.vlaanderen.be/publicaties/stem-agenda-2030-stem-competenties-voor-een-toekomst-en-missiegericht-beleid>

A.2. State-of-the-art

A.2.1. Tekstvereenvoudiging

De voorbije tien jaar is artificiële intelligentie (AI) sterk verder ontwikkeld. Vasista (2022) benadrukt dat de toename in kennis voor nieuwe toepassingen zorgde. Tekstvereenvoudiging vloeide hier uit voort. Momenteel bestaan er al robuuste toepassingen die teksten kunnen vereenvoudigen, zoals Resoomer², Paraphraser³ en Prepostseo⁴. Binnen het kader van tekstvereenvoudiging is er bestaande documentatie beschikbaar waar onderzoekers het voordeel van toegankelijkheid aanhalen, maar volgens Gooding (2022) ontbreken deze toepassingen de extra noden die scholieren met dyslexie in de derde graad middelbaar onderwijs vereisen.

Shardlow (2014) haalt aan dat het algemene doel van tekstvereenvoudiging is om ingewikkelde bronnen toegankelijker te maken. Het zorgt voor verkorte teksten zonder de kernboodschap te verliezen. Siddharthan (2014) haalt verder aan dat tekstvereenvoudiging op één van drie manieren gebeurt. Er is conceptuele vereenvoudiging waarbij documenten naar een compacter formaat worden getransformeerd. Daarnaast is er uitgebreide modificatie die kernwoorden aanduidt door gebruik van redundantie. Als laatste is er samenvatting die documenten verandert in kortere teksten met alleen de topische zinnen. Met deze concepten zijn ontwikkelaars volgens Siddharthan (2014) in staat om ingewikkelde woorden te vervangen door eenvoudigere synoniemen of zinnen te verkorten zodat ze sneller leesbaar zijn.

Tekstvereenvoudiging behoort tot de zijtak van *Natural Language Processing* (NLP) in AI. NLP omvat methodes om menselijke teksten om te zetten in tekst voor machines. Documenten vereenvoudigen met NLP kan volgens Chowdhary (2020) op twee manieren: extraherend of abstraherend. Bij extraherende vereenvoudiging worden zinnen gelezen zoals ze zijn neergeschreven. Vervolgens bewaart een document de belangrijkste taalelementen om de tekst te kunnen hervormen. Deze vorm van tekstvereenvoudiging komt volgens (Sciforce, 2020) het meeste voor. Daarnaast is er abstraherende vereenvoudiging waarbij de kernboodschap wordt bewaard. Met de kernboodschap wordt er een nieuwe zin opgebouwd. Volgens het onderzoek van Chowdhary (2020) heeft deze vorm potentieel, maar het zit nog in de kinderschoenen.

A.2.2. Noden van scholieren met dyslexie

Het experiment van Franse wetenschappers

Gala en Ziegler (2016) illustreert dat manuele tekstvereenvoudiging schoolteksten toegankelijker

maakt voor kinderen met dyslexie. Dit deden ze door simpelere synoniemen en

²<https://resoomer.com/nl/>

³<https://www.paraphraser.io/nl/tekst-samenvatting>

⁴<https://www.prepostseo.com/tool/nl/text-summarizer>

zinsstructuren te gebruiken. Tien kinderen werden opgenomen in het experiment, variërend van 8 tot 12 jaar oud. Verwijswoorden werden vermeden en woorden kort gehouden. De resultaten waren veelbelovend. Het leestempo lag hoger en de kinderen maakten minder leesfouten. Ook bleek er geen verlies van begrip in de tekst bij geteste kinderen. Resultaten van de studie werden gebundeld voor de mogelijke ontwikkeling van een AI-tool.

De visuele weergave van tekst beïnvloedt de leessnelheid bij scholieren met dyslexie. Zo haalt het onderzoek van Rello e.a. (2012b) tips aan waarmee teksten en documenten rekening moeten houden bij scholieren met dyslexie in de derde graad middelbaar onderwijs. Het gaat over speciale lettertypes, spreiding tussen woorden en het gebruik van inzoomen op aparte zinnen. Het onderzoek haalt verder aan dat teksten voor deze unieke noden aanpassen tijdervend is en daarmee tekst vereenvoudiging door AI een revolutionaire oplossing kan bieden. De Universiteit van Kopenhagen is met bovenstaande idee aan de slag gegaan. Onderzoekers Bingel e.a. (2018) hebben gratis software ontwikkeld, genaamd Hero⁵, om tekstvereenvoudiging voor scholieren in het middelbaar onderwijs met dyslexie te automatiseren. De software bestudeert met welke woorden de gebruiker moeite heeft, en vervangt die door simpelere alternatieven. Hero bevindt zich nu in beta-vorm en wordt enkel in het Engels en Deens ondersteund. Als alternatief is er Readable⁶. Dit is een Engelstalige AI-toepassing dat zinnen beoordeeld met leesbaarheidsformules.

Roldós (2020) haalt aan dat NLP in de laatste decennia volop in ontwikkeling is, maar ontwikkelaars botsen nog op uitdagingen. Het gaat om zowel interpretatie- als dataproblemen bij AI-modellen. Het onderzoek haalt twee punten aan. Allereerst is het voor een machine moeilijk om de context van homoniemen te achterhalen. Bijvoorbeeld bij het woord ‘bank’ is het niet duidelijk voor de machine of het gaat over de geldinstelling of het meubel. Daarnaast zijn synoniemen een probleem voor tekstverwerking.

Het onderzoek van Sciforce (2020) haalt aan dat het merendeel van NLP-toepassingen Engelstalige invoer gebruikt. Niet-Engelstalige toepassingen zijn zeldzaam. De opkomst van AI technologieën die twee datasets gebruiken, biedt een oplossing voor dit probleem. De software vertaalt eerst de oorspronkelijke tekst naar de gewenste taal, voordat de tekst wordt herwerkt. Hetzelfde onderzoek bewijst dat het vertalen van gelijkaardige talen, zoals Duits en Nederlands, een minimaal verschil opleverd. Volgens Plavén-Sigray e.a. (2017) houden onderzoekers zich vaak in hun eigen taalbubbel, wat negatieve gevolgen heeft voor de leesbaarheid van een wetenschappelijk artikel. Bovendien vormt de stijgende trend van het gebruik aan acroniemen Barnett en Doubleday (2020) een extra hindernis. Donato e.a. (2022) haalt aan dat onbegrijpelijke literatuur, waaronder studiemateriaal geschreven door de docent

⁵<https://beta.heroapp.ai/>

⁶<https://readable.com/>

en online wetenschappelijke artikelen, één van de redenen is waarom scholieren met dyslexie in het middelbaar onderwijs van richting veranderen.

A.2.3. Huidige toepassingen

Vlaanderen heeft weinig zicht op de geïmplementeerde AI software in scholen. Dit werd vastgesteld door (Martens e.a., 2021a), een samenwerking tussen de Vlaamse universiteiten en overheid voor AI. vergeleken met andere Europese landen, maakt België het minst gebruik van leerling-georiënteerde hulpmiddelen. Degenen die wel gebruikt worden, zijn vooral online leerplatformen voor zelfstandig werken. Ook maakt België amper gebruik van beschikbare software die de leermethoden en -noden van leerlingen evalueert (Martens e.a., 2021b).

Verhoeven (2023) haalt aan dat AI-toepassingen zoals ChatGPT, Google Bard en Bing AI kunnen helpen om routinematisch werk te verminderen in het onderwijs. Echter haalt Deckmyn (2021) aan dat GPT-3, het model van ChatGPT, sterker staat in het maken van Engelstalige teksten vergeleken met Nederlandstalige teksten. De databank waar het GPT-3 model mee is getraind, bestaat uit 92% Engelstalige woorden, terwijl er 0,35% Nederlandse woorden aanwezig zijn in dezelfde databank. Ontwikkelaars moeten de evolutie van deze modellen opvolgen, voordat er Nederlandstalige toepassingen mee worden gemaakt.

A.2.4. Ontwikkelen met AI

Python staat bovenaan de lijst van programmeertalen voor NLP-toepassingen. Volgens het onderzoek van Thangarajah (2019) is dit te wijten aan de eenvoudige syntax, kleine leercurve en grote beschikbaarheid van kant-en-klare bibliotheken. Wiskundige berekeningen of statistische analyses kunnen worden uitgevoerd met één lijn code. Malik (2022) haalt de twee meest voorkomende aan, namelijk NLTK⁷ en Spacy⁸. Deep Martin⁹ bouwt verder op het onderzoek van Shardlow (2014) naar een pipeline voor lexicale vereenvoudiging. Deep Martin maakt gebruik van custom transformers om invoertekst te converteren naar een vereenvoudigde versie van de teksthoud.

Voor Germaanse talen zijn er enkele datasets en word embeddings beschikbaar die de complexiteit van woorden bijhouden. Zo zijn er in de Duitse taal Klexikon¹⁰ en TextComplexityDE¹¹. Een onderzoek van Suter e.a. (2016) bouwde een rule-based NLP-model met 'Leichte Sprache', wat een dataset is met eenvoudige Duitstalige zinsconstructies. Nederlandstalige datasets zijn in schaarse hoeveelheden beschikbaar, waardoor het vertalen uit een Germaanse taal is hier een optie.

Volgens Garbacea e.a. (2021) is het belangrijk dat AI-ontwikkelaars niet alleen aan-

⁷<https://www.nltk.org/>

⁸<https://spacy.io/>

⁹<https://github.com/chrislemke/deep-martin>

¹⁰<https://github.com/dennlinger/klexikon>

¹¹<https://github.com/babaknaderi/TextComplexityDE>

dacht besteden aan het aanpassen van woorden en zinnen, maar ook aan de gebruiker meegeven waarom iets is aangepast. De onderzoekers wijzen op twee ethische aspecten. Eerst moet de toepassing duidelijk aangeven waarom een woord of zin is aangepast. Het model moet de moeilijkheidsgraad van de woorden of zinnen bewijzen. Iavarone e.a. (2021) beschrijft een methode met regressiemodellen om de moeilijkheidsgraad te bepalen door een gemiddeld moeilijkheidspercentage per zin te berekenen. Daarnaast benadrukt Garbacea e.a. (2021) het belang van het markeren van de complexere delen van een tekst. Hiervoor haalt hetzelfde onderzoek methoden aan zoals *lexical of deep learning*.

Er is een tactvolle aanpak nodig om een vereenvoudigde tekst met AI te beoordelen. De studie van Swayamdipta (2019) haalt aan dat er extra nood is aan NLP-modellen waarbij de tekst zijn kernboodschap behoudt. Samen met Microsoft Research bouwden ze NLP-modellen die gericht waren op de bewaring van zinsstructuur en -context door *scaffolded learning*. Hiervoor maakten de onderzoekers gebruik van een voorspellingsmethode die de positie van woorden en zinnen in een document beoordeelde. De Flesch-Kincaid leesbaarheidstest is volgens Readable (2021) een alternatieve manier om vereenvoudigde tekstinhoud te beoordelen, zonder de nood aan *pre-trained* modellen. Deze score kan eenvoudig worden berekend met de Python-library *textstat*¹².

A.3. Methodologie

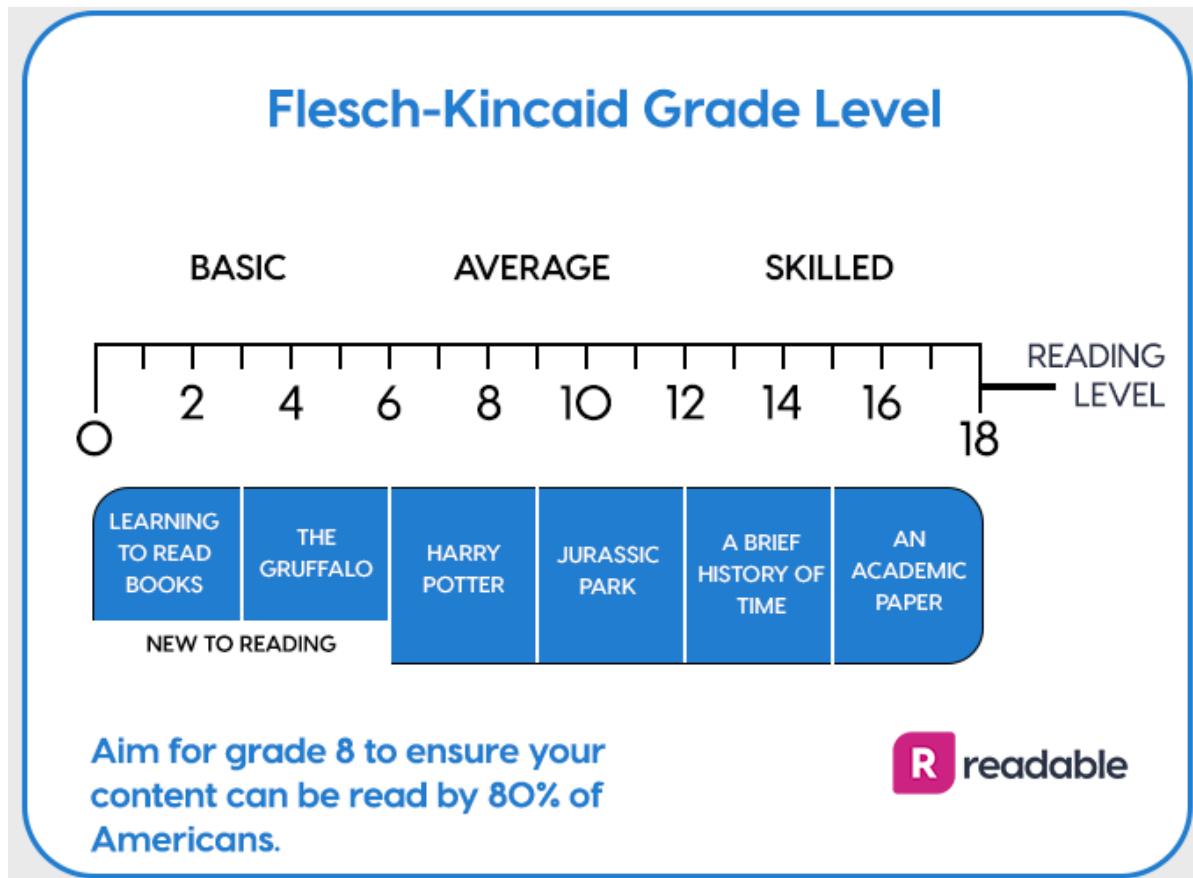
Een *mixed-methods* onderzoek toont aan hoe toepassingen automatisch een wetenschappelijke artikel kunnen vereenvoudigen, gericht op scholieren met dyslexie in de derde graad middelbaar onderwijs. Het onderzoek houdt vijf grote fasen in. De eerste fase is het proces van geautomatiseerde tekstvereenvoudiging beschrijven. Dit gebeurt via een grondige studie van vakliteratuur en wetenschappelijke teksten. Ook blogs van experts komen hier aan bod. Na het verwerven van de nodige inzichten wordt er een verklarende tekst opgesteld.

De tweede fase bestaat uit het analyseren van wetenschappelijke werken over de bewezen voordelen van tekstvereenvoudiging bij scholieren met dyslexie van de derde graad middelbaar onderwijs. Hiervoor zijn geringe theissen beschikbaar, die zorgvuldigheid vragen tijdens interpretatie. De resulterende tekst bevat de voordelen samen met hun wetenschappelijke onderbouwing.

De derde fase is opnieuw een beschrijving. Hier worden de valkuilen bij taalverwerking met AI-software nagegaan. Deze fase van het onderzoek brengt mogelijke nadelen en tekortkomingen van AI-software bij tekstvereenvoudiging aan het licht. Dit gebeurt aan de hand van een technische uitleg.

De vierde fase omvat een toelichting over beschikbare AI toepassingen voor tekstvereenvoudiging. Aan de hand van een veldonderzoek op het internet en bij bedrijven wordt een longlist opgesteld van beschikbare toepassingen voor tekstver-

¹²<https://pypi.org/project/textstat/>



Figuur (A.1)
(Readable, 2021)

eenvoudiging in het middelbaar onderwijs. Met een requirementsanalyse wordt er een shortlist opgesteld van software. Het toetsen van verschillende tools wordt ook betrokken in deze fase. De shortlist vormt de basis voor de ontwikkeling van een prototype voor geautomatiseerde en adaptieve tekstvereenvoudiging.

De vijfde en laatste fase van het onderzoek bestaat uit het testen en beoordelen van gekozen AI-toepassingen voor tekstvereenvoudiging. In dit experiment proberen scholieren met dyslexie in de derde graad middelbaar onderwijs de shortlisted AI toepassingen en het prototype uit. Het doel van het experiment is om de effectiviteit en gebruikersvriendelijkheid van deze toepassingen te beoordelen. Na een grondige analyse wordt er met de resultaten bepaalt of de toepassingen aan de unieke noden van een scholier met dyslexie in de derde graad middelbaar onderwijs voldoen om wetenschappelijke artikelen te vereenvoudigen voor scholieren in het middelbaar onderwijs.

A.4. Verwacht resultaat, conclusie

Er wordt verwacht dat de huidige softwareoplossingen voor tekstvereenvoudiging onvoldoende aansluiten bij de noden van scholieren met dyslexie in de derde graad middelbaar onderwijs. Het prototype is moeilijk af te stemmen op de specifieke noden van deze doelgroep. Ontwikkelaars die werken met bestaande modellen moeten *custom transformers* inzetten om bevredigende resultaten te krijgen. Bovendien ontbreken er Nederlandstalige word embeddings die de complexiteit van elk woord bijhouden en aan kant-en-klare modellen die de inhoud van wetenschappelijke artikelen kunnen vereenvoudigen. Word embeddings uit een Germaanse taal gebruiken, gevolgd door vertaling naar het Nederlands is wel een aanvaardbaar alternatief.

B

Referentieteksten: Richtlijnen

Het onderzoek achterhaalt hoe scholieren met dyslexie in de derde graad middelbaar onderwijs ondersteund kunnen worden bij het intensief lezen van een wetenschappelijk artikel. De ondersteuning wordt aangeboden in de vorm van tekstvereenvoudiging met AI. Tekstvereenvoudiging omvat het lexicaal en syntactisch vereenvoudigen, alsook het samenvatten van de kerngedachte per hoofdstuk. Om tekstvereenvoudiging met AI te testen, moeten handmatig en automatisch vereenvoudigde teksten met elkaar worden vergeleken.

De opdracht voor deze bijdrage is het manueel vereenvoudigen van een gekregen wetenschappelijk artikel. Dit wetenschappelijk artikel is zes pagina's (voorpagina uitgesloten) lang. Het doelpubliek voor dit vereenvoudigd artikel zijn scholieren met dyslexie in de derde graad ASO/TSO middelbaar onderwijs. Concreet zou dit een artikel moeten zijn dat tijdens een STEM-les wordt gegeven aan deze doelgroep. Op pagina 2 vindt u tekstvereenvoudigingstechnieken terug. Deze aanpassingen hebben een beneficieel effect op scholieren met dyslexie een wetenschappelijk artikel bij het intensief lezen van wetenschappelijke teksten. U dient deze gekregen aanpassingen te volgen voor deze bijdrage. De beschreven technieken en elementen dienen in de manuele vereenvoudigde tekst terug te vinden zijn.

Op basis van de richtlijnen op pagina 2 worden dezelfde instructies aan een AI-model gegeven. Met de richtlijnen en de door u handmatig vereenvoudigde tekst kan het onderzoek evalueren of AI-taalmodellen capabel zijn om manuele tekstvereenvoudigingstechnieken, specifiek voor scholieren met dyslexie, toe te passen op wetenschappelijke artikelen. De tekst dat een AI-model vereenvoudigd wordt afgetoetst op basis van bestaande metrieken en de kenmerken van uw vereenvoudigde versie.

Voor de vereenvoudigde versie van het artikel moet u als taaldocent of auteur geen rekening houden met marges, lettertypes of spatiëring. Deze aanpassingen mogen, maar enkel de tekstuele inhoud van het gekregen document wordt in het experiment opgenomen. Een Word-document of PDF-document is voldoende. Daarnaast moet er ook geen rekening worden gehouden met de afbeeldingen in het artikel.

Aanpassingen die niet op pagina 2 omschreven zijn om de tekst eenvoudiger te maken, zijn vrijblijvend. Indien deze aanpassing volgens u een meerwaarde biedt, dan moet de werkwijze voor de start van het document kort beschreven worden. De aanpassing moet eenmalig bovenaan het document worden vermeld. Bijvoorbeeld: 'De zin werd gesplitst omdat deze langer is dan tien woorden.' Zo kunnen wij bij het onderzoek rekening houden met deze extra handeling. Het AI-model wordt dan met deze extra parameter in het achterhoofd beoordeeld.

Namens mijn promotor, copromotors en mezelf wil ik u hartelijk bedanken voor uw interesse in dit onderzoek.

B.0.1. Lexicale vereenvoudiging

- Een moeilijk woord achterhalen gebeurt op basis van intuïtie en inschatting van de doelgroep. De woordenschat die zelden voorkomt in de dagelijkse lees- en schrijftaal van STEM-vakken voor scholieren tussen 16 en 18 jaar oud, moet worden aangepast. Vakjargon die al in de tweede graad ASO en TSO aan bod is gekomen, mag behouden blijven.
- Een woord dat langer is dan achttien letters, wordt als moeilijk beschouwd en moet vervangen worden door een korter (en eenvoudiger) alternatief.
- Acroniemen worden voluit geschreven.
- Vervang een moeilijk woord in het artikel door slechts één synoniem. Bijvoorbeeld, als het woord 'adhesief' wordt vervangen door 'klevend', gebruik dan geen andere synoniemen voor 'klevend' in de rest van het artikel.
- Indien een woord geen eenvoudiger synoniem heeft, mag het woord kort worden uitgelegd. Dit kan tussen ronde haakjes, of in een aparte zin. Bijvoorbeeld: "Ik voelde me melancholisch." wordt aangepast naar "Ik had een diep gevoel van droefheid en verlies."
- Vermijd het directe overnemen van percentages indien deze voorkomen in het artikel. Vervang dit door benamingen zoals 'een kwart', 'de helft'.

B.0.2. Syntactische vereenvoudiging

- Te lange zinnen worden opgebroken of gesplitst. De zinnen in het vereenvoudigde artikel zijn hoogstens tien woorden lang.
- Verwijswoorden zoals 'zij', 'hun' of 'hij' worden naar namen veranderd. Bijvoorbeeld voornamen of entiteitsnamen (bijvoorbeeld Nationale Bank).
- Tangconstructies worden vervangen. Dit kan door de bijzin naar het begin of het einde van een zin te plaatsen, de zin te splitsen in twee kortere zinnen of door het onderwerp en de persoonsvorm dichter bij elkaar te plaatsen door minder informatie tussenin te plaatsen.
- Voorzetseluitdrukkingen en samengestelde werkwoorden worden vervangen indien mogelijk. Indien er geen eenvoudigere alternatieven zijn, mogen deze onaangepast blijven.

B.0.3. Samenvatten

- Het vereenvoudigde artikel volgt dezelfde structuur en chronologische volgorde zoals dat van het oorspronkelijk artikel. Iedere hoofdstuk in het wetenschappelijk artikel is hoogstens twee paragrafen lang. Per paragraaf zijn er hoogstens vijf zinnen.

- Het vereenvoudigd artikel is hoogstens 500 woorden lang.
- Citeren mag indien deze zinnen aan de bovenstaande criteria (lexicale en syntactische vereenvoudiging) voldoen.
- Het gebruik van opsommingen of *bullet-points* wordt aangemoedigd.
- De bronvermelding wordt overgenomen. De referentie gebeurt zoals die uit het oorspronkelijke document (Vancouver) en mag direct overgenomen worden: '[4]' blijft '[4]'.

B.1. Specifieke richtlijnen voor AI

De kerngedachte van iedere paragraaf moet terug te vinden zijn in de vereenvoudigde tekst. Na de tekstvereenvoudiging moeten de volgende vragen in hoogstens twee paragrafen beantwoord kunnen worden:

- **Inleiding:** Wat is het doel van dit onderzoek? Uit welk eerder onderzoek of uit welke probleemstelling vloeide dit onderzoek voort?
- **Socio-technische ontwikkeling:** Welke drie technische ontwikkelingen worden aangehaald in het onderzoek? Wat zijn de sociotechnische ontwikkelingen die het traditionele controle- en handhavingskader onder druk zetten als gevolg van de opkomst van algoritmische surveillance in het politiewerk?
- **Juridisch kader:** Wat zijn de tekortkomingen van het huidige juridisch kader en de controle-instrumenten die momenteel worden ingezet voor de verwerking van gegevens door middel van AI, en biedt het recente voorstel van de EU voor een AI-wet voldoende bescherming van grondrechten en handhavingsmechanismen?
- **Herdenken van algoritmische surveillance-controle:** Hoe kan de visie van Ubuntufilosofie en relationele ethiek bijdragen aan een herziening? Hoe kan relationele controle helpen bij het beschermen van kwetsbare groepen tegen schendingen van mensenrechten door algoritmische surveillance?
- **Concrete stappen:** Welke concrete stappen omtrent ethiek worden er aangehaald? Hoe kan relationele controle helpen bij het herdenken van controlemechanismen en rekening houden met sociaal-technische ontwikkelingen zoals asymmetrische machtsrelaties en de toenemende macht van technologiebedrijven?
- **Conclusies:** Wat besluiten de onderzoekers? Indien verder onderzoek vereist is, naar welk onderzoek kijken ze specifiek uit?

B.2. Specifieke richtlijnen voor A2

Het doel is om de kerngedachte van iedere paragraaf in de vereenvoudigde tekst terug te kunnen vinden, alsook een antwoord moet kunnen geven op de onderstaande vragen per sectie. Enkel de doorlopende tekst moet worden vereenvoudigd, dus geen extra uitleg over de grafieken en visualisaties. Daarnaast moet de vereenvoudigde tekst een antwoord kunnen geven op de vragen in hoogstens vier paragrafen beantwoord kunnen worden:

- **Inleiding:** Wat is de probleemstelling voor dit onderzoek? Welk doel heeft deze bijdrage? Opmerking: uitzonderlijk moet deze sectie tot hoogstens één paragraaf worden samengevat.
- **Beleidsaanpak:**
 - Welke economische problemen zijn er ontstaan als gevolg van de oliecrisis en invoerconcurrentie in Nederland en België?
 - Wat waren de belangrijkste beleidswijzigingen? Welke gevolgen waren er?
 - Wat zijn de belangrijkste verschillen tussen de Nederlandse en Belgische economie en wat zijn de belangrijkste uitdagingen waar deze landen momenteel voor staan?
 - Hoe verschillen de aanpak en uitgaven van de overheid in België en Nederland en wat zijn de gevolgen daarvan voor hun economieën en overheidsfinanciën?
- **Competitiviteit**
 - Welke factoren hebben geleid tot het verschil in economische prestaties tussen Nederland en België, en wat is de rol van de werkzaamheidsgraad in deze verschillen?
 - Wat zijn de belangrijkste redenen voor het verschil in groeiprestaties tussen Nederland en België, en welke factoren spelen hierbij een rol, met name op het gebied van arbeidsmarkt, innovatie en ondernemerschap?
 - Welke observaties worden er gemaakt over ondernemerschap en innovatie in Nederland en België?
 - Wat is het verband tussen de inkomende en uitgaande buitenlandse directe investeringen als percentage van het BBP en de internationale化atie van bedrijven in Nederland en België?
- **Structurele evoluties in beide landen:**
 - Welke factoren hebben geleid tot de de-industrialisatie in België en Nederland en welke impact heeft dit gehad op de werkgelegenheid en productiviteit in beide landen?

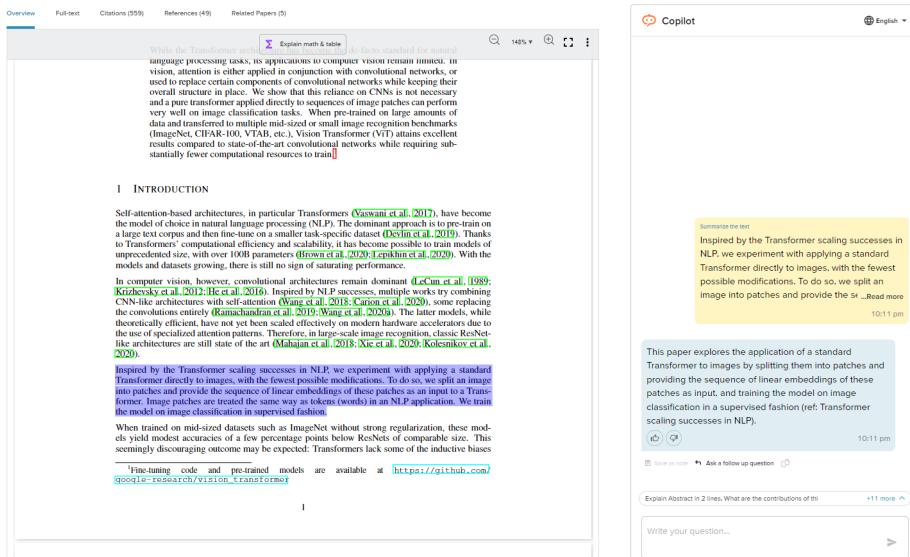
- Hoe beïnvloedt de ongelijke groei tussen de dienstensector en de industriële sector de productiviteit en de economische groei in België?
- Hoe verschilt de dynamiek van de drie gewesten in België met betrekking tot de economische bevoegdheden en de neerwaartse convergentiekrachten in de EU?

Opmerking: Er worden hier drie vragen gesteld, maar u mag nog steeds hoogstens vier paragrafen gebruiken om deze sectie samen te vatten en te vereenvoudigen.

- **Conclusies:** Wat besluiten de onderzoekers? Indien verder onderzoek vereist is, naar welk onderzoek kijken ze specifiek uit? Opmerking: uitzonderlijk moet deze sectie tot hoogstens twee paragrafen worden samengevat.

C

Figures: Requirementsanalyse



Figuur (C.1)

Informatie opvragen van een wetenschappelijk artikel met SciSpace

kan je deze tekst samenvatten: <https://www.frontiersin.org/articles/10.3389/fpsyg.2021.726882/full>

✓ Zoeken naar <https://www.frontiersin.org/articles/10.3389/fpsyg.2021.726882/full>
 ✓ Antwoorden voor u genereren...

Dit is een mogelijke samenvatting van de tekst:
 De tekst gaat over hoe het leren lezen de luisterende hersenen verandert. De auteurs bespreken hoe het lezen twee netwerken in de hersenen activeert: een fonologisch netwerk dat zich richt op de klanken van woorden en een orthografisch netwerk dat zich richt op de vormen van letters. Ze stellen voor dat het lezen van een vaardigheid is die geleerd kan worden door audio-visuele oefeningen en longitudinale studies te gebruiken om de veranderingen in de hersenen en het gedrag te volgen terwijl kinderen leren lezen [1]. Ze suggereren dat deze benadering kan helpen om beter te begrijpen hoe het lezen invloed heeft op andere cognitieve processen, zoals taal, geheugen en aandacht.

Is dit wat je zocht?

Meer informatie: 1. frontiersin.org 2. frontiersin.org 3. frontiersin.org 2 van 8

wat kan je mij meer vertellen over het aanleren van deze vaardigheid in de visuele cortex?

✓ Zoeken naar 'Learning to read visual cortex'
 ✓ Antwoorden voor u genereren...

Volgens sommige onderzoeken verandert het leren lezen de visuele cortex door een specifieke hersenplaats in de linker occipito-temporale cortex te activeren, die de **visuele woordvormgebied** (VWFA) wordt genoemd. Het herkennen van woorden activeren dezelfde stimuli in het geleerde schrift en helpt bij het herkennen van woorden [1] [2]. Het leren lezen recyclent dus de visuele corticale netwerken zonder ze te vernietigen of te verdriegen [2] [3]. Het VWFA is ook betrokken bij andere taken die visuele expertise vereisen, zoals gezichtsherkenning of schaakspel [4].

Heb je nog meer vragen?

Meer informatie: 1. science.org 2. science.org 3. pubmed.ncbi.nlm.nih.gov 3 van 8

Figuur (C.2)

Tekstvereenvoudiging via de link van een wetenschappelijk artikel met Bing Chat

Color code	Description
Black	Words in Black don't change between the two versions.
Green	Words in Green mean they have been translated adequately.
Purple	Words in Purple display a further explanation in foot notes.
Olive	Words in Olive contain two or more possible meanings (a tooltip is provided for these words, place the mouse cursor on top of olive words to see possible meanings).
Blue	Words in Blue are recognized in Wikipedia (normally names, places, people, organizations, etc.).
Orange	Words in Orange are not currently available in Basic English.
Red	Words in Red are names, special terms or not recognized by the translating tool.

Note : Double click on any word to add it to your personal dictionary.

Input Text

Artificial intelligence has been applied more into occupations by companies and individuals. However, the effects within the benefits are both imaginable and unpredictable. Sexual discrimination in jobs is also a debatable topic. The purpose of this paper is to combine the topics of both AI and sexual discrimination and discuss their effects in the job field in the future. Automation, big data and the algorithm applied in the job field would be some of the points to discover. To briefly summarize, automation is the use of machines and computers that reduces human intervention. Big data is a collection of data from various sources, it is related to AI because the more data input into AI the better it becomes. Since AI absorbs the information and learns from them. AI algorithm takes the data input and uses mathematics and logic to produce the output. [1] Gender discrimination in AI not only reflects the pre-existing biases in the society, but it could also reinforce them through automation, hiring system and decision making. This paper is not totally against the use of AI but advocates that artificial intelligence should be used in a more careful, gender responsible way to reduce sexual discrimination in the job field.

Simplified

artificial intelligence¹ has been made a request more into work by companies and beings, however, the effects within the gets help are both idea forming and not able to say before hand. sex caused decision making in regular work is also an about which argument is possible thing talked of. The purpose of this paper is to trading group the interests of both AI and sex caused decision making and have a discussion about their effects in the regular work field in the future. automation², greatly sized facts and the algorithm³ is applied in the job field. automation is the use of machines and computers that reduces human intervention. Big data is a group of facts from different starting points, it is related to AI because the more facts input into AI the better it becomes. Since AI takes up the news given and learns from them. AI algorithm⁴ takes the facts input and uses mathematics and tests, reasoning to produce the out put. [1] sex statement decision making in AI not only gives back (light, heat, sound) the in existence beforehand has a tendency in a certain direction in the society, but it could also make stronger them through automation², getting use of person for money system and decision making. This paper is not totally against the use of AI but Advocates⁴ that artificial intelligence¹ should be used in a more careful, sex statement responsible way to get changed to other form sex caused decision making in the regular work field.

Menu ▾

artificial intellig... science that gives great weight to ways of making come into existence intelligent machines that work and have reactions like those of man. Continue reading.

automation² the technology of making machines, instruments, process, and the like go through a certain train of operations without further impulse or control from outside after being started. Continue reading

algorithm³ a word coming from the name of the expert in mathematics /Al-Khwarizmi@who (780-850ac), used to give the way to work out or solve points to be answered. Continue reading.

Advocates⁴ A barrister or solicitor representing a party in a hearing before a Court. Continue reading.

Figuur (C.3)

Illustratie van de tekstanalyse bij Simplish na een tekstvereenvoudiging.

Problem: Many people read very little, because it's too hard to do. Look at the text that we inserted into the yellow box below, starting with, *The waif tried to save...* Can you understand it? Millions of people can't, because the words are too hard.

Settings

Sample Original:
Rewordify.com is a sublime web site that expedites learning in myriad ways. It helps with reading betterment, and it invites discourse on more topics.

Sample Output:
Rewordify.com is an amazing web site that speeds up learning in many ways. It helps with reading (improvement/ positive change), and it invites intelligent conversation about more topics.

Display mode: ?

- Reward hard words; click/tap to see original
- Don't reward words; click/tap to see definitions
- Display hard word and easier word inline
- Display original / rewordified in two columns
- Display original with vocabulary column

Rewordifying level: ?

- Easiest
- Level 1 (default)
- Level 2
- Level 3
- Level 4
- Hardest

Highlighting mode: ?

- Yellow/purple
- Green/light red
- Blue/orange
- Underline
- None

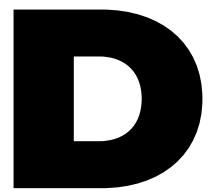
Help improve learning outcomes

Rewordify.com's amazing features have helped millions of people read billions of words more easily.

Save & close

Figuur (C.4)

Illustratie van de tekstanalyse bij Rewordify.



Code: Ontwikkeling Van Het Prototype

```
1 import subprocess, io, os, pypandoc
2 from datetime import date
3 import zipfile
4
5 markdown_file = "saved_files/file.md"
6 zip_filename = 'saved_files/simplified_docs.zip'
7 pdf_file = "saved_files/output.pdf"
8 docx_file = "saved_files/output.docx"
9 DATE_NOW = str(date.today())
10
11 class Creator():
12     def create_header(self, title, margin, fontsize, chosen_font, chosen_title_font,
13                       word_spacing, type_spacing):
14         with open(markdown_file, 'w', encoding='utf-8') as f:
15             f.write("---\n")
16             f.write(f"title: {title}\n")
17             f.write(f"mainfont: {chosen_font}.ttf\n")
18             f.write(f"titlefont: {chosen_title_font}.ttf\n")
19             f.write(f'date: {DATE_NOW}\n')
20             f.write(f'document: article\n')
21             f.write(f'geometry: margin={margin}cm\n')
22             f.write(f'fontsize: {fontsize}pt\n')
23             f.write('header-includes:\n')
24             f.write(f'- \spacekip={word_spacing}cm\n')
25             f.write(f'- \\usepackage{{setspace}}\n')
26             f.write(f'- \\{type_spacing}\n')
27             f.write("---\n")
28
29     def generate\_glossary(self, list):
30         with open(markdown_file, 'a', encoding='utf-8') as f:
```

```

31     f.write("---\n")
32     f.write("# Woordenlijst\n")
33     f.write("| Woord | Soort | Definitie |\n")
34     f.write("| --- | --- | --- |\n")
35     for word in list.keys():
36         f.write(f"| {word} | {list[word]['type']}| {list[word]['definition']}| \n"
37     )
38
39 """"""
40 def generate_summary(self, full_text):
41     with open(markdown_file, 'a', encoding="utf-8", errors="surrogateescape") as f:
42         for key in full_text.keys():
43             title = str(key).replace('\n', ' ')
44             text = full_text[key]
45             f.write('\n\n')
46             f.write(f'## {title}')
47             f.write('\n\n')
48             f.write(" ".join(text))
49             f.write('\n\n')
50
51 def generate_summary_w_summation(self, full_text):
52     with open(markdown_file, 'a', encoding="utf-8", errors="surrogateescape") as f:
53         for key in full_text.keys():
54             title = str(key).replace('\n', ' ')
55             text = full_text[key][0].split(' | ')
56             f.write('\n\n')
57             f.write(f'## {title}')
58             for sentence in text:
59                 f.write('\n\n')
60                 f.write(f'* {sentence}')
61                 f.write('\n\n')
62
63
64 def create_pdf(self, title, margin, list, full_text, fonts, word_spacing,
65               type_spacing, summation):
66     if title is not None:
67         self.create_header(title=title, margin=margin, fontsize=14, chosen_font=
68                           fonts[0], chosen_title_font=fonts[1], word_spacing=word_spacing, type_spacing=
69                           type_spacing)
70     else:
71         self.create_header(title='Vereenvoudigde tekst', margin=0.5, fontsize=14,
72                           chosen_font=fonts[0], chosen_title_font=fonts[1], word_spacing=word_spacing,
73                           type_spacing=type_spacing)
74
75 """GLOSSARY"""
76 if len(list) != 0:
77     self.generate_glossary(list=list)
78
79 """SUMMARY"""
80 if summation:

```

```

76     self.generate_summary_w_summation(full_text=full_text)
77 else:
78     self.generate_summary(full_text=full_text)
79
80 """FILE_CREATION"""
81 pypandoc.convert_file(source_file=markdown_file, to='docx', outputfile=
82 docx_file, extra_args=['-M2GB', '+RTS', '-K64m', '-RTS'])
83 pypandoc.convert_file(source_file=markdown_file, to='pdf', outputfile=
84 pdf_file, extra_args=['--pdf-engine=xelatex'])
85 with zipfile.ZipFile(zip_filename, 'w') as myzip:
86     myzip.write(pdf_file)
87     myzip.write(docx_file)

```

Listing D.1: Writer-klasse omvattende de code om dynamische PDF- en Word-documenten te genereren.

```

1 /* Adjectieven uitfilteren op basis van span-tag */
2 function removeAdjectives() {
3     const elements = document.querySelectorAll(".adj");
4     elements.forEach(function (element) {
5         element.remove();
6     });
7 }
8
9 /* Checkboxes */
10 const nouns = document.getElementById('noun-show');
11 const verbs = document.getElementById('verb-show');
12 const adjs = document.getElementById('adj-show');
13
14 nouns.addEventListener('change', function () {
15     if (this.checked) {
16         const color = document.getElementById('colorForNouns').value;
17         console.log(color);
18         const elements = document.querySelectorAll("span.noun");
19         elements.forEach(function (element) {
20             element.style.color = color;
21         });
22     } else {
23         const elements = document.querySelectorAll("span.noun");
24         elements.forEach(function (element) {
25             element.style.color = "black";
26         });
27     }
28 });
29
30 verbs.addEventListener('change', function () {
31     if (this.checked) {
32         const color = document.getElementById('colorForVerbs').value;
33         const elements = document.querySelectorAll("span.verb, span.aux");
34         console.log(color, elements[0])
35         elements.forEach(function (element) {
36             element.style.color = color;

```

```

37     });
38 } else {
39     const elements = document.querySelectorAll("span.verb, span.aux");
40     elements.forEach(function (element) {
41         element.style.color = "black";
42     });
43 }
44 });
45
46
47 adjs.addEventListener('change', function () {
48     if (this.checked) {
49         const color = document.getElementById('colorForAdjs').value;
50         const elements = document.querySelectorAll("span.adj");
51         elements.forEach(function (element) {
52             element.style.color = color;
53         });
54     } else {
55         const elements = document.querySelectorAll("span.adj");
56         elements.forEach(function (element) {
57             element.style.color = "black";
58         });
59     }
60 });

```

Listing D.2: De toegepaste scripts voor het verwijderen van adjektieven en togglen van type woorden.

```

1 document.addEventListener("DOMContentLoaded", () => {
2     const spans = document.querySelectorAll(".verb, .adj, .noun, .aux");
3     spans.forEach((span) => {
4         span.addEventListener("click", async (event) => {
5             const radioButton = document.querySelector("#explainWords");
6             if (radioButton && !radioButton.checked) {
7                 return;
8             }
9             let leftSideTag = span.closest("p");
10            let rightSideTag = leftSideTag.nextElementSibling;
11            sentence_of_origin = span.closest(".sentence");
12
13            var context = "";
14            for (const child of sentence_of_origin.children) {
15                context = context + " " + child.textContent;
16            }
17            const word = event.target.textContent;
18            const response = await fetch(`http://localhost:5000/look-up-word`, {
19                method: "POST",
20                headers: { "Content-Type": "application/json" },
21                body: JSON.stringify({ word: word, sentence: context }),
22            });
23            result = await response.json();
24

```

```

25     if (result.result == "error") {
26         alert("Incorrect API key provided: " + result.word);
27     } else {
28         var pos_tag = result.result.split(' | ')[0];
29         var definition = result.result.split(' | ')[1];
30         let table = document.querySelector(".table-glossary");
31         let newRow = table.insertRow(-1);
32         let cell1 = newRow.insertCell(0);
33         let cell2 = newRow.insertCell(1);
34         let cell3 = newRow.insertCell(2);
35         cell1.innerHTML = result.word;
36         cell2.innerHTML = 'Bijvoeglijk naamwoord';
37         cell3.innerHTML = definition.lower;
38     }
39 });
40 });
41 });
42
43 /* --- */
44 async function syntacticSimplification() {
45     var selectedText = window.getSelection().toString();
46     if (selectedText == "" || selectedText == null) {
47         alert('Markeer de tekst die u wilt vereenvoudigen.');
48         return;
49     }
50
51     var dazzle = document.querySelector(".dazzle");
52     var p = document.createElement("p");
53     var p2 = document.createElement("p");
54     var text = document.createTextNode("Zinsbouw vereenvoudigen... ");
55     var prompt = document.createTextNode("... ");
56     p.appendChild(prompt);
57     dazzle.appendChild(p);
58     p2.appendChild(text);
59     dazzle.appendChild(p2);
60     const response = await fetch(`http://localhost:5000/simplify`, {
61         method: "POST",
62         headers: { "Content-Type": "application/json" },
63         body: JSON.stringify({ text: selectedText, key: "sc" }),
64     });
65     result = await response.json();
66     prompt.nodeValue = JSON.stringify(result.prompt);
67     text.nodeValue = JSON.stringify(result.result);
68 }
69
70 /* --- */
71 function insertAfter(newNode, existingNode) {
72     existingNode.parentNode.insertBefore(newNode, existingNode.nextSibling);
73 }
74
75 /* --- */

```

```

76 document.addEventListener("DOMContentLoaded", () => {
77   const spans = document.querySelectorAll(".sentence");
78   spans.forEach((span) => {
79     span.addEventListener("click", async (event) => {
80       const radioButton = document.querySelector("#simplifySentences"); // get
81       reference to radio button
82       if (radioButton && !radioButton.checked) {
83         return;
84       }
85       sentence_of_origin = span.closest(".sentence");
86       var context = "";
87       for (const child of sentence_of_origin.children) {
88         context = context + " " + child.textContent;
89       }
90       const response = await fetch(`http://localhost:5000/simplify`, {
91         method: "POST",
92         headers: { "Content-Type": "application/json" },
93         body: JSON.stringify({ text: context, key: "sc" }),
94       });
95       result = await response.json();
96       var p = document.createElement("p");
97       newNode = document.createTextNode(JSON.stringify(result.result));
98       p.append(newNode);
99       insertAfter(p, span);
100      sentence_of_origin.innerHTML = "<del>" + context + "</del>";
101      parent = sentence_of_origin.parent;
102    });
103  });
104
105 async function personalizedSimplification() {
106   var selectedText = window.getSelection().toString();
107   if (selectedText == "" || selectedText == null) {
108     alert('Markeer de tekst die u wilt vereenvoudigen.');
109     return;
110   }
111
112   let checkedValues = [];
113   let checkboxes = document.querySelectorAll(
114     '.personalisation input[type="checkbox"]'
115   );
116
117   checkboxes.forEach((checkbox) => {
118     if (checkbox.checked) {
119       checkedValues.push(checkbox.name);
120     }
121   });
122
123   if (checkedValues.length == 0) {
124     alert('Duidt één van de onderstaande opties aan.');
125     return;

```

```

126 }
127
128 var selectedChoices = checkedValues;
129 var prompt = document.createTextNode("Gepersonaliseerde tekst ophalen...");
130 var text = document.createTextNode("...");  

131 var dazzle = document.querySelector(".dazzle");
132 var p = document.createElement("p");
133 var p2 = document.createElement("p");
134
135 p.appendChild(prompt);
136 dazzle.appendChild(p);
137 p2.appendChild(text);
138 dazzle.appendChild(p2);
139
140 const response = await fetch(`http://localhost:5000/personalized-simplify`, {
141   method: "POST",
142   headers: { "Content-Type": "application/json" },
143   body: JSON.stringify({ text: selectedText, choices: selectedChoices }),
144 });
145
146 result = await response.json();
147
148 array = result.result;
149 console.log(array);
150
151 if (array.length > 1) {
152   prompt.nodeValue = JSON.stringify(result.prompt);
153   const ul = document.createElement("ul");
154   array.forEach((item) => {
155     const li = document.createElement("li");
156     li.textContent = item;
157     ul.appendChild(li);
158   });
159   text.remove();
160   p2.appendChild(ul);
161 } else {
162   prompt.nodeValue = JSON.stringify(result.prompt);
163   text.nodeValue = JSON.stringify(result.result[0]);
164 }
165 }
166
167 async function askGPT() {
168   var selectedText = window.getSelection().toString();
169   if (selectedText == "" || selectedText == null) {
170     alert('Markeer de tekst die u wilt vereenvoudigen.');
171     return;
172   }
173
174   var promptText = window.prompt(
175     "Wat wilt u doen met de geselecteerde tekst? Schrijf hier de prompt...""
176 );

```

```
177
178 var fullPrompt = promptText + "///\n" + selectedText;
179 var prompt = document.createTextNode("Gepersonaliseerde tekst ophalen...");  

180 var text = document.createTextNode("");  

181 var dazzle = document.querySelector(".dazzle");  

182 var p = document.createElement("p");  

183 var p2 = document.createElement("p");  

184  

185 p.appendChild(prompt);  

186 dazzle.appendChild(p);  

187 p2.appendChild(text);  

188 dazzle.appendChild(p2);  

189  

190 const response = await fetch(  

`http://localhost:5000/personalized-simplify-custom-prompt`,  

{  

  method: "POST",  

  headers: { "Content-Type": "application/json" },  

  body: JSON.stringify({ text: selectedText, prompt: fullPrompt }),  

}  

);  

197 );  

198  

199 result = await response.json();  

200 console.log(result);  

201 prompt.nodeValue = JSON.stringify(result.prompt);  

202 text.nodeValue = JSON.stringify(result.result);  

203 }  

204  

205 async function emptyChat() {  

206   const dazzleDiv = document.querySelector(".dazzle");  

207   dazzleDiv.innerHTML = "";  

208 }  

209  

210 window.onload = async function () {  

211   /* --- */  

212   var url = `http://localhost:5000/get-settings-user`;  

213   const response = await fetch(url, { method: "POST" });  

214   var result = await response.json();  

215   document.body.style.fontSize = result.fontSize + "px";  

216   document.body.style.fontFamily = result.fontSettings;  

217   document.body.style.backgroundColor = result.favcolor;  

218   document.body.style.lineHeight = result.lineHeight + "cm";  

219   document.body.style.wordSpacing = result.wordSpacing + "cm";  

220   document.body.style.textAlign = result.textAlignment;  

221  

222   /* --- */  

223   var url = "http://localhost:5000/get-session-keys";  

224   const session_keys_response = await fetch(url, { method: "POST" });  

225   result = await session_keys_response.json();  

226  

227   let missing_keys = [];
```

```

228
229 if (result.hf_api_key === undefined) {
230   missing_keys.push("HuggingFace");
231   document.querySelector("#simple-syntactic-simplification").style.display =
232     "none";
233 }
234
235 if (result.gpt3 === undefined) {
236   missing_keys.push("GPT-3");
237   document.querySelector("#prompt-synt-simplification").style.display =
238     "none";
239   document.querySelector("#personalized-synt-simplification").style.display =
240     "none";
241   document.querySelector(".personalisation").style.display = "none";
242 }
243
244 if (missing_keys.length > 0){
245   alert("Sleutel(s) voor " + missing_keys.join(" & ") + " ontbreken.");
246 }
247 };

```

Listing D.3: De toegepaste scripts voor enkel het scholierencomponent.

```

1 /* --- */
2 const checkbox = document.querySelector("#personalizedSummary");
3 const fieldsets = document.querySelectorAll(".personalized");
4 checkbox.addEventListener("change", () => {
5   fieldsets.forEach((fieldset) => {
6     if (checkbox && checkbox.checked) {
7       fieldset.style.display = "block";
8     } else {
9       fieldset.style.display = "none";
10    }
11  });
12 });
13
14 /* Add word to glossary */
15 var checkboxAddWordToGlossary = document.getElementById(
16 "checkboxAddWordToGlossary"
17 );
18 checkboxAddWordToGlossary.addEventListener("change", function () {
19   if (checkboxAddWordToGlossary && checkboxAddWordToGlossary.checked) {
20     const words = document.querySelectorAll(
21       "span.verb, span.noun, span.aux, span.verb, span.adj"
22     );
23     words.forEach((w) => {
24       w.addEventListener("click", async (event) => {
25         if (checkboxAddWordToGlossary.checked) {
26           var pTag = event.target;
27           sentence_of_origin = w.closest("span.sentence");
28           var context = "";

```

```

29         for (const child of sentence_of_origin.children) {
30             context = context + " " + child.textContent;
31         }
32         console.log(context);
33         var textarea = document.getElementById("glossaryList");
34         pTag.style.backgroundColor = "black";
35         pTag.style.color = "white";
36         pTag.style.fontWeight = "bold";
37         textarea.value += pTag.innerHTML + ":" + context + "\n";
38         console.log(textarea.value);
39     }
40   });
41 });
42 }
43 });
44
45 /* Deleting sentences */
46 const checkboxDeleteSents = document.getElementById("checkboxDeleteSents");
47 checkboxDeleteSents.addEventListener("change", function () {
48   if (checkboxDeleteSents && checkboxDeleteSents.checked) {
49     const sentences = document.querySelectorAll(".sentence");
50     sentences.forEach((span) => {
51       span.addEventListener("click", async (event) => {
52         if (checkboxDeleteSents.checked) span.remove();
53       });
54     });
55   }
56 });
57
58 /* Tekst toevoegen */
59 function addTextToTextArea() {
60   const fullTextBox = document.querySelector(".left-container").innerHTML;
61   var textarea = document.getElementById("fullText");
62   textarea.value = fullTextBox;
63   document.getElementById("summarize-with-presets-button").disabled = false;
64 }
65
66 function makeTitle(button) {
67   const titleText = document.getElementById("title").value;
68   const newTitle = document.createElement("h3");
69   newTitle.innerText = titleText;
70   const titleContainer = document.getElementById("title").parentElement;
71   titleContainer.replaceChild(newTitle, document.getElementById("title"));
72   button.parentNode.removeChild(button);
73 }
74
75 function deleteTitle(button) {
76   var parent = button.parentNode;
77   parent.parentNode.removeChild(parent);
78 }
79

```

```

80 window.onload = async function () {
81     /* --- */
82     var url = `http://localhost:5000/get-settings-user`;
83     const response = await fetch(url, { method: "POST" });
84     var result = await response.json();
85     document.body.style.fontSize = result.fontSize + "px";
86     document.body.style.fontFamily = result.fontSettings;
87     document.body.style.backgroundColor = result.favcolor;
88     document.body.style.lineHeight = result.lineHeight + "cm";
89     document.body.style.wordSpacing = result.wordSpacing + "cm";
90     document.body.style.textAlign = result.TextAlign;
91
92     /* --- */
93     var url = "http://localhost:5000/get-session-keys";
94     const session_keys_response = await fetch(url, { method: "POST" });
95     result = await session_keys_response.json();
96
97     let missing_keys = [];
98
99     if (result.hf_api_key === undefined) {
100         missing_keys.push("HuggingFace");
101         document.getElementById("huggingface").innerHTML =
102             "Geen HuggingFace sleutel werd opgegeven. ";
103         document.getElementById("huggingface").style.color = "red";
104     } else {
105         document.getElementById("huggingface").innerHTML =
106             "HuggingFace API-sleutel:\t" + result.hf_api_key;
107     }
108
109     if (result.gpt3 === undefined) {
110         missing_keys.push("GPT-3");
111         document.getElementById("gpt3").innerHTML =
112             "Geen GPT-3 sleutel werd opgegeven.";
113         document.getElementById("gpt3").style.color = "red";
114     } else {
115         document.getElementById("gpt3").innerHTML =
116             "GPT-3 API-sleutel: " + result.gpt3;
117     }
118
119     if (missing_keys.length > 0) {
120         alert("Sleutel(s) voor " + missing_keys.join(" & ") + " ontbreken.");
121     }
122 };

```

Listing D.4: De toegepaste scripts voor enkel het lerarencomponent.

```

1 FROM python:3.8-slim-buster
2
3 WORKDIR /app
4
5 COPY requirements.txt requirements.txt

```

```

6
7 RUN apt-get update && apt-get install -y pandoc texlive-xetex texlive poppler-
  utils
8
9 RUN pip3 install -r requirements.txt \
10 && python3 -m spacy download nl_core_news_md \
11 && python3 -m spacy download en_core_web_md
12
13 COPY . .
14
15 CMD [ "python3" , "-m" , "flask" , "run" , "--host=0.0.0.0" , "--port=5000"]

```

Listing D.5: Dockerfile voor het prototype.

```

1 @echo off
2
3 cd web-app
4 docker stop text-application-prototype
5 docker rm text-application-prototype
6
7 docker rmi text-app
8
9 docker build -t text-app .
10 docker run --name text-application-prototype --network webapp_simplification -d -p
    5000:5000 text-app

```

Listing D.6: Script voor het opstarten van de Docker-container voor Windows-gebruikers

```

1 #!/bin/sh
2
3 cd web-app || exit
4 docker stop text-application-prototype
5 docker rm text-application-prototype
6
7 docker rmi text-app
8
9 docker build -t text-app .
10 docker run --name text-application-prototype --network webapp_simplification -d -p
    5000:5000 text-app

```

Listing D.7: Script voor het opstarten van de Docker-container voor Unix-gebruikers

Bibliografie

- Althunayyan, S. & Azmi, A. (2021). Automated Text Simplification: A Survey. *ACM Computing Surveys*, 54, Article no. 43. <https://doi.org/10.1145/3442695>
- Ball, P. (2017). It's not just you: science papers are getting harder to read. *Nature*.
- Barnett, A. & Doubleday, Z. (2020). Meta-Research: The growth of acronyms in the scientific literature (P. Rodgers, Red.). *eLife*, 9, e60080.
- Belpaeme, T., Kennedy, J., Ramachandran, A., Scassellati, B. & Tanaka, F. (2018). Social robots for education: A review. *Science robotics*, 3(21), eaat5954.
- Bezem, A. & Lugthart, M. (2016). Visuele Disfunctie een onzichtbare belemmering bij lezen, spelling en concentratie. <https://beeldenbrein.nl/>
- Bilici, Ş. (2021). Sequence labeling.
- Bingel, J., Paetzold, G. & Søgaard, A. (2018). Lexi: A tool for adaptive, personalized text simplification. *Proceedings of the 27th International Conference on Computational Linguistics*, 245–258.
- Binz, M. & Schulz, E. (2023). Using cognitive psychology to understand GPT-3. *Proceedings of the National Academy of Sciences*, 120(6).
- Bonte, M. (2020). *Bestaat Dyslexie?: En is het een relevante vraag?* uitgeverij SWP.
- Botelho, F. H. F. (2021). Accessibility to digital technology: Virtual barriers, real opportunities [PMID: 34951832]. *Assistive Technology*, 33(supl), 27–34. <https://doi.org/10.1080/10400435.2021.1945705>
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language Models are Few-Shot Learners.
- Bulté, B., Sevens, L. & Vandeghinste, V. (2018). Automating lexical simplification in Dutch. *Computational Linguistics in the Netherlands Journal*, 8, 24–48. <https://clijournal.org/clij/article/view/78>
- Canning, Y., Tait, J., Archibald, J. & Crawley, R. (2000). Cohesive Generation of Syntactically Simplified Newspaper Text. In P. Sojka, I. Kopeček & K. Pala (Red.), *Text, Speech and Dialogue* (pp. 145–150). Springer Berlin Heidelberg.
- Cao, M. (2022). A Survey on Neural Abstractive Summarization Methods and Factual Consistency of Summarization.
- Charlesworth Author Services. (2021). <https://www.cwauthors.com/article/How-to-write-about-complex-scientific-concepts-in-simple-accessible-language>
- Chowdhary, K. (2020). *Fundamentals of Artificial Intelligence*. Springer, New Delhi.

- Coster, W. & Kauchak, D. (2011). Learning to Simplify Sentences Using Wikipedia. *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, 1–9. <https://aclanthology.org/W11-1601>
- Crevits, H. (2022, maart 13). *Kwart van bedrijven gebruikt artificiële intelligentie: Vlaanderen bij beste leerlingen van de klas* (Persbericht). Vlaamse Overheid Departement Economie, Wetenschap en Innovatie.
- Dandekar, N. (2016). How to use machine learning to find synonyms. <https://medium.com/@nikhilbd/how-to-use-machine-learning-to-find-synonyms-6380c0c6106b>
- Dapaah, J. & Maenhout, K. (2022, juli 8). *Iedereen heeft boter op zijn hoofd* (D. Standaard, Red.). https://www.standaard.be/cnt/dmf20220607_97763592
- De Craemer, J., Van Beeumen, L., Cooreman, A., Moonen, A., Rottier, J., Wagemakers, I. & Mardulier, T. (2018). Aan de slag met voorleessoftware op school. Een gids met 8 vragen en antwoorden. <https://onderwijs.vlaanderen.be/nl/onderwijspersoneel/van-basis-tot-volwassenenonderwijs/lespraktijk/ict-in-de-klas/voorleessoftware-voor-leerlingen-met-leesbeperkingen/aan-de-slag-met-voorleessoftware-op-school>
- De Meyer, I., Janssens, R. & Warlop, N. (2019). Leesvaardigheid van 15-jarigen in Vlaanderen: Overzicht van de eerste resultaten van PISA2018. <https://data-onderwijs.vlaanderen.be/documenten/bestand.ashx?id=12265>
- Deckmyn, D. (2021, maart 19). *Robot schrijft mee De Standaard* (D. Standaard, Red.). https://www.standaard.be/cnt/dmf20210319_05008561
- Donato, A., Muscolo, M., Arias Romero, M., Caprì, T., Calarese, T. & Olmedo Moreno, E. M. (2022). Students with dyslexia between school and university: Post-diploma choices and the reasons that determine them. An Italian study. *Dyslexia*, 28(1), 110–127.
- DuBay, W. H. (2004). The principles of readability. *Online Submission*.
- Eisenstein, J. (2019). *Introduction to Natural Language Processing*. MIT Press. <https://books.google.be/books?id=72yuDwAAQBAJ>
- Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R. & Radev, D. (2020). SumEval: Re-evaluating Summarization Evaluation.
- Gala, N. & Ziegler, J. (2016). Reducing lexical complexity as a tool to increase text accessibility for children with dyslexia. *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*, 59–66.
- Galliussi, J. e.a. (2020). Inter-letter spacing, inter-word spacing, and font with dyslexia-friendly features: testing text readability in people with and without dyslexia. *Annals of Dyslexia*, 70, 141–152.
- Garbacea, C., Guo, M., Carton, S. & Mei, Q. (2021). Explainable Prediction of Text Complexity: The Missing Preliminaries for Text Simplification. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*

- (Volume 1: Long Papers), 1086–1097. <https://doi.org/10.18653/v1/2021.acl-long.88>
- Garg, H. (2022). Using GPT-3 for education: Use cases. <https://indiaai.gov.in/article/using-gpt-3-for-education-use-cases>
- Ghesquière, P. (2018). *Als leren pijn doet: Kinderen met een leerstoornis opvoeden en begeleiden*. Acco.
- Gooding, S. (2022). On the Ethical Considerations of Text Simplification. *Ninth Workshop on Speech and Language Processing for Assistive Technologies (SLPAT-2022)*, 50–57. <https://doi.org/10.18653/v1/2022.spat-1.7>
- Gooding, S. & Kochmar, E. (2019). Complex word identification as a sequence labelling task. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1148–1153.
- Greg, B., Atty, E., Elie, G., Joane, J., Logan, K., Lim, R., Luke, M. & Michelle, P. (2023). Introducing chatgpt and Whisper Apis. <https://openai.com/blog/introducing-chatgpt-and-whisper-apis>
- Gupta, S. & Gupta, S. K. (2019). Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121, 49–65. <https://doi.org/https://doi.org/10.1016/j.eswa.2018.12.011>
- Hahn, U. & Mani, I. (2000). The Challenges of Automatic Summarization. *Computer*, 33, 29–36. <https://doi.org/10.1109/2.881692>
- Hartley, J. (1999). From Structured Abstracts to Structured Articles: A Modest Proposal. *Journal of Technical Writing and Communication*, 29(3), 255–270. <https://doi.org/10.2190/3RWW-A579-HC8W-6866>
- Harwell, D. (2023). Tech's hottest new job: Ai whisperer. no coding required. <https://www.washingtonpost.com/technology/2023/02/25/prompt-engineers-techs-next-big-job/>
- Hayes, D. P. (1992). The growing inaccessibility of science. <https://www.nature.com/articles/356739a0>
- Hern, A. (2023). TechScape: Will meta's massive leak democratise AI – and at what cost? <https://www.theguardian.com/technology/2023/mar/07/techscape-meta-leak-llama-chatgpt-ai-crossroads>
- Hollenkamp, J. (2020). Summary and analysis of Scientific Research Articles - San Jose State ... <https://www.sjsu.edu/writingcenter/docs/handouts/Summary%20and%20Analysis%20of%20Scientific%20Research%20Articles.pdf>
- Hsu, W.-T., Lin, C.-K., Lee, M.-Y., Min, K., Tang, J. & Sun, M. (2018). A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss.
- Huang, S., Wang, R., Xie, Q., Li, L. & Liu, Y. (2019). An Extraction-Abstraction Hybrid Approach for Long Document Summarization. *2019 6th International Conference on Behavioral, Economic and Socio-Cultural Computing (BESC)*, 1–6.

- Hubbard, K. E. & Dunbar, S. D. (2017). Perceptions of scientific research literature and strategies for reading papers depend on academic career stage. *PLOS ONE*, 12(12), 1–16.
- Iavarone, B., Brunato, D. & Dell'Orletta, F. (2021). Sentence Complexity in Context. *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, 186–199. <https://doi.org/10.18653/v1/2021.cmcl-1.23>
- IBM. (2022). IBM Global AI Adoption Index 2022. <https://www.ibm.com/downloads/cas/GVAGA3JP>
- Iredale, G. (2022). An overview of tokenization algorithms in NLP. <https://101blockchains.com/tokenization-nlp/>
- Iskender, N., Polzehl, T. & Möller, S. (2021). Reliability of Human Evaluation for Text Summarization: Lessons Learned and Challenges Ahead. *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, 86–96. <https://aclanthology.org/2021.humeval-1.10>
- Jiang, R. K. (2023). Prompt engineering : Deconstructing and managing intention. <https://www.linkedin.com/pulse/prompt-engineering-deconstructing-managing-intention-jiang/>
- Jones, R., Colusso, L., Reinecke, K. & Hsieh, G. (2019). r/science: Challenges and Opportunities in Online Science Communication. *CHI '19: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–14. <https://doi.org/10.1145/3290605.3300383>
- Jurafsky, D., Martin, J., Norvig, P. & Russell, S. (2014). *Speech and Language Processing*. Pearson Education. <https://books.google.be/books?id=Cq2gBwAAQBAJ>
- Kandula, S., Curtis, D. & Zeng-Treitler, Q. (2010). A semantic and syntactic text simplification tool for health content. *AMIA annual symposium proceedings*, 2010, 366.
- Khan, A. (2014). A Review on Abstractive Summarization Methods. *Journal of Theoretical and Applied Information Technology*, 59, 64–72.
- Khurana, D., Koli, A., Khatter, K. & Singh, S. (2022). Natural Language Processing: State of The Art, Current Trends and Challenges. *Multimedia Tools and Applications*, 82, 25–27.
- Kraft, M. A. (2020). Interpreting Effect Sizes of Education Interventions. *Educational Researcher*, 49(4), 241–253. <https://doi.org/10.3102/0013189X20912798>
- Li, C. (2022). OpenAI's GPT-3 language model: A technical overview. <https://lambdalabs.com/blog/demystifying-gpt-3>
- Li, J., Sun, A., Han, J. & Li, C. (2018). A Survey on Deep Learning for Named Entity Recognition.
- Lin, H. & Bilmes, J. (2010). Multi-document summarization via budgeted maximization of submodular functions. *Human Language Technologies: The 2010*

- Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 912–920.
- Linderholm, T., Everson, M. G., van den Broek, P., Mischinski, M., Crittenden, A. & Samuels, J. (2000). Effects of Causal Text Revisions on More- and Less-Skilled Readers' Comprehension of Easy and Difficult Texts. *Cognition and Instruction*, 18(4), 525–556.
- Lissens, F., Asmar, M., Willems, D., Van Damme, J., De Coster, S., Demeestere, E., Maes, R., Baccarne, B., Robaeyst, B., Duthoo, W. & Desoete, A. (2020). Het stopt nooit...De impact van dyslexie en/of dyscalculie op het welbevinden en studeren van (jong)volwassenen en op de transitie naar de arbeidsmarkt: een bundeling van Vlaamse pilootstudies.
- Liu, Q., Kusner, M. J. & Blunsom, P. (2020). A Survey on Contextual Embeddings.
- Malik, R. S. (2022, juli 4). *Top 5 NLP Libraries To Use in Your Projects* (T. Al, Red.). <https://towardsai.net/p/l/top-5-nlp-libraries-to-use-in-your-projects>
- Martens, M., De Wolf, R. & Evens, T. (2021a). *Algoritmes en AI in de onderwijscontext: Een studie naar de perceptie, mening en houding van leerlingen en ouders in Vlaanderen*. Kenniscentrum Data en Maatschappij. Verkregen 30 maart 2022, van <https://data-en-maatschappij.ai/publicaties/survey-onderwijs-2021>
- Martens, M., De Wolf, R. & Evens, T. (2021b, juni 28). *School innovation forum 2021*. Kenniscentrum Data en Maatschappij. Verkregen 1 april 2022, van <https://data-en-maatschappij.ai/nieuws/school-innovation-forum-2021>
- McCombes, S. (2022). How to write A summary: Guide amp; examples. <https://www.scribbr.com/working-with-sources/how-to-summarize/>
- McDonald, R. (2007). A study of global inference algorithms in multi-document summarization. *Advances in Information Retrieval: 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007. Proceedings* 29, 557–564.
- McFarland, A. (2023). What is prompt engineering in AI amp; Why It Matters. <https://www.unite.ai/what-is-prompt-engineering-in-ai-why-it-matters/>
- McKeown, K., Klavans, J. L., Hatzivassiloglou, V., Barzilay, R. & Eskin, E. (1999). Towards multidocument summarization by reformulation: Progress and prospects.
- McNutt, M. (2014). Reproducibility. *Science*, 343(6168), 229–229. <https://doi.org/10.1126/science.1250475>
- Menzli, A. (2023). Tokenization in NLP: Types, challenges, examples, tools. <https://neptune.ai/blog/tokenization-in-nlp>
- Miszczak, P. (2023). Prompt engineering: The ultimate guide 2023 [GPT-3 amp; chat-gpt]. <https://businessolution.org/prompt-engineering/>
- Mottesi, C. (2023). GPT-3 vs. Bert: Comparing the two most popular language models. <https://blog.invgate.com/gpt-3-vs-bert>

- Nallapati, R., Zhai, F. & Zhou, B. (2017). SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1). <https://doi.org/10.1609/aaai.v31i1.10958>
- Nandhini, K. & Balasundaram, S. (2013). Improving readability through extractive summarization for learners with reading difficulties. *Egyptian Informatics Journal*, 14(3), 195–204.
- Nenkova, A. & Passonneau, R. (2004). Evaluating Content Selection in Summarization: The Pyramid Method. *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, 145–152.
- Niemeijer, A., Frederiks, B., Riphagen, I., Legemaate, J., Eefsting, J. & Hertogh, C. (2010). Ethical and practical concerns of surveillance technologies in residential care for people with dementia or intellectual disabilities: an overview of the literature. *Psychogeriatrics*, 22(7), 1129–1142. <https://doi.org/10.1017/S1041610210000037>
- Onderwijsinspectie Overheid Vlaanderen. (2020). <https://www.vlaanderen.be/publicaties/begrijpend-leesonderwijs-in-de-basisscholen-kwaliteitsvol-sterke-en-zwakke-punten-van-de-huidige-praktijk>
- OnderwijsVlaanderen. (2023). Voorleessoftware voor Leerlingen met Leesbeperkingen. <https://onderwijs.vlaanderen.be/voorleessoftware-voor-leerlingen-met-leesbeperkingen>
- Paetzold, G. & Specia, L. (2016). SemEval 2016 Task 11: Complex Word Identification. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 560–569. <https://doi.org/10.18653/v1/S16-1085>
- Pain, E. (2016). How to (seriously) read a scientific paper. <https://www.science.org/content/article/how-seriously-read-scientific-paper>
- Plavén-Sigray, P., Matheson, G. J., Schiffler, B. C. & Thompson, W. H. (2017). Research: The readability of scientific texts is decreasing over time (S. King, Red.). *eLife*, 6, e27725.
- Poel, M., Boschman, E. & op den Akker, R. (2008). A Neural Network Based Dutch Part of Speech Tagger [<http://eprints.ewi.utwente.nl/14662>; 20th Benelux Conference on Artificial Intelligence, BNAIC 2008, BNAIC ; Conference date: 30-10-2008 Through 31-10-2008]. In A. Nijholt, M. Pantic, M. Poel & H. Hondorp (Red.), *BNAIC 2008* (pp. 217–224). Twente University Press (TUP).
- Premjith, P., John, A. & Wilscy, M. (2015). Metaheuristic Optimization Using Sentence Level Semantics for Extractive Document Summarization, 347–358. https://doi.org/10.1007/978-3-319-26832-3_33
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. e.a. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 9.

- Rani, R. & Kaur, B. (2021). The TEXT SUMMARIZATION AND ITS EVALUATION TECHNIQUE. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(1), 745–752.
- Readable. (2021). *Flesch Reading Ease and the Flesch Kincaid Grade Level*. <https://readable.com/readability/flesch-reading-ease-flesch-kincaid-grade-level/>
- Rello, L. & Baeza-Yates, R. (2013). Good fonts for dyslexia. *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2013*.
- Rello, L., Baeza-Yates, R., Dempere-Marco, L. & Saggion, H. (2013). Frequent Words Improve Readability and Short Words Improve Understandability for People with Dyslexia.
- Rello, L., Baeza-Yates, R. & Saggion, H. (2013). The Impact of Lexical Simplification by Verbal Paraphrases for People with and without Dyslexia. 7817, 501–512.
- Rello, L. & Baeza-Yates, R. A. (2015). How to present more readable text for people with dyslexia. *Universal Access in the Information Society*, 16, 29–49.
- Rello, L. & Bigham, J. (2017). Good Background Colors for Readers: A Study of People with and without Dyslexia, 72–80.
- Rello, L., Kanvinde, G. & Baeza-Yates, R. (2012a). Layout Guidelines for Web Text and a Web Service to Improve Accessibility for Dyslexics. *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*.
- Rello, L., Kanvinde, G. & Baeza-Yates, R. (2012b). Layout Guidelines for Web Text and a Web Service to Improve Accessibility for Dyslexics. *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*.
- Ribas, J. (2023). Building the new bing. <https://www.linkedin.com/pulse/building-new-bing-jordi-ribas/>
- Ribeiro, E., Ribeiro, R. & de Matos, D. M. (2018). A Study on Dialog Act Recognition using Character-Level Tokenization.
- Rijkhoff, J. (2022). Tekst Inkorten?: 9 tips om Je Teksten korter Te Maken. <https://dialoogtrainers.nl/tekst-inkorten-tips/>
- Rivero-Contreras, M., Engelhardt, P. E. & Saldaña, D. (2021). An experimental eye-tracking study of text adaptation for readers with dyslexia: effects of visual support and word frequency. *Annals of Dyslexia*, 71, 170–187.
- Roldós, I. (2020, december 22). *Major Challenges of Natural Language Processing (NLP)*. MonkeyLearn. Verkregen 1 april 2022, van <https://monkeylearn.com/blog/natural-language-processing-challenges/>
- Roose, K. (2023). Don't ban chatgpt in schools. teach with it. <https://www.nytimes.com/2023/01/12/technology/chatgpt-schools-teachers.html>

- Ruelas Inzunza, E. (2020). Reconsidering the Use of the Passive Voice in Scientific Writing. *The American Biology Teacher*, 82, 563–565. <https://doi.org/10.1525/abt.2020.82.8.563>
- Santana, V., Oliveira, R., Almeida, L. & Baranauskas, M. C. (2012). Web accessibility and people with dyslexia: A survey on techniques and guidelines. W4A 2012 - International Cross-Disciplinary Conference on Web Accessibility. <https://doi.org/10.1145/2207016.2207047>
- Sciforce. (2020, februari 4). Biggest Open Problems in Natural Language Processing. Verkregen 1 april 2022, van <https://medium.com/sciforce/biggest-open-problems-in-natural-language-processing-7eb101ccfc9>
- Shardlow, M. (2013). A Comparison of Techniques to Automatically Identify Complex Words. *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, 103–109. <https://aclanthology.org/P13-3015>
- Shardlow, M. (2014). A Survey of Automated Text Simplification. *International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing 2014*, 4(1). <https://doi.org/10.14569/SpecialIssue.2014.040109>
- Siddharthan, A. (2006). Syntactic Simplification and Text Cohesion. *Research on Language and Computation*, 4(1), 77–109. <http://oro.open.ac.uk/58888/>
- Siddharthan, A. (2014). A survey of research on text simplification. *ITL - International Journal of Applied Linguistics*, 165, 259–298.
- Sikka, P. & Mago, V. (2020). A Survey on Text Simplification. *CoRR*, *abs/2008.08612*. <https://arxiv.org/abs/2008.08612>
- Simon, J. (2021). Large language models: A new moore's law? <https://huggingface.co/blog/large-language-models>
- Sleuwaegen, L. (2022). Nederland versus België: verschillen in economischenbsp; dynamiek en beleid. <https://feb.kuleuven.be/research/les/pdf/LES%202022%20-%2020197.pdf>
- Snow, C. (2010). Academic Language and the Challenge of Reading for Learning About Science. *Science* (New York, N.Y.), 328, 450–2.
- Sohom, G., Ghosh; Dwight. (2019). *Natural Language Processing Fundamentals*. Packt Publishing. <https://medium.com/analytics-vidhya/natural-language-processing-basic-concepts-a3c7f50bf5d3>
- Stajner, S. (2021). Automatic Text Simplification for Social Good: Progress and Challenges, 2637–2652. <https://doi.org/10.18653/v1/2021.findings-acl.233>
- Strubell, E., Ganesh, A. & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP.

- Suleiman, D. & Awajan, A. (2020). Deep Learning Based Abstractive Text Summarization: Approaches, Datasets, Evaluation Measures, and Challenges. *Mathematical Problems in Engineering*, 2020.
- Suter, J., Ebling, S. & Volk, M. (2016). Rule-based Automatic Text Simplification for German.
- Swayamdipta, S. (2019, januari 22). *Learning Challenges in Natural Language Processing*. Verkregen 1 april 2022, van <https://www.microsoft.com/en-us/research/video/learning-challenges-in-natural-language-processing/>
- Tanya Goyal, G. D., Junyi Jessy Li. (2022). News Summarization and Evaluation in the Era of GPT-3. *arXiv preprint*.
- Thangarajah, V. (2019). Python current trend applications-an overview.
- Tops, W., Callens, M., Brysbaert, M. & Schouten, E. L. (2018). *Slagen met Dyslexie in Het Hoger Onderwijs*. Owl Press.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models.
- Van Brakel, R. (2022). De controle op het gebruik van algoritmische surveillance-onder druk? Een exploratie door de lens van de relationele ethiek. *Tijdschrift voor Mensenrechten*, 2022(1), 23–28.
- van der Meer, C. (2022). Dyslexie hebben is Niet Zo Raar: Lezen is iets heel onnatuurlijks. <https://www.demorgen.be/beter-leven/dyslexie-hebben-is-niet-zo-raar-lezen-is-iets-heel-onnatuurlijks~bc608101/>
- Vasista, K. (2022). Evolution of AI Design Models. *Central Asian Journal of Theoretical and Applied Science*, 3(3), 1–4.
- Verhoeven, W. (2023, februari 8). *Applaus voor de studenten die ChatGPT gebruiken* (Trends, Red.). https://trends.knack.be/economie/bedrijven/applaus-voor-de-studenten-die-chatgpt-gebruiken/article-opinion-1934277.html?cookie_check=1676034368
- Verma, P. & Verma, A. (2020). A review on text summarization techniques. *Journal of scientific research*, 64(1), 251–257.
- White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J. & Schmidt, D. C. (2023). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT.
- Xu, W., Callison-Burch, C. & Napoles, C. (2015). Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3, 283–297.
- Zeng, Q., Kim, E., Crowell, J. & Tse, T. (2005). A Text Corpora-Based Estimation of the Familiarity of Health Terminology. In J. L. Oliveira, V. Maojo, F. Martín-Sánchez & A. S. Pereira (Red.), *Biological and Medical Data Analysis* (pp. 184–192). Springer Berlin Heidelberg.

- Zhang, M., Riecke, L. & Bonte, M. (2021). Neurophysiological tracking of speech-structure learning in typical and dyslexic readers. *Neuropsychologia*, 158, 107889.
- Zhou, W., Ge, T., Xu, K., Wei, F. & Zhou, M. (2019). BERT-based Lexical Substitution. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3368–3373. <https://doi.org/10.18653/v1/P19-1328>