Scholieren met dyslexie van de derde graad middelbaar onderwijs ondersteunen bij het begrijpend 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
Tweede examenperiode

Departement IT en Digitale Innovatie .

HO GENT

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 Vercruysse 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, en ook 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 de derde graad van het middelbaar onderwijs bij het begrijpend lezen van wetenschappelijke artikelen. Gepersonaliseerde *manual text simplification* (MTS) is een gekende en bewezen techniek die deze scholieren helpt met hun leesbegrip. De techniek vraagt echter veel tijd en energie en is daardoor minder inzetbaar. Artificiële intelligentie maakt de automatisering van het proces mogelijk en vermindert de werkdruk bij leraren en scholieren. 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 stelt het onderzoek de volgende onderzoeksvraag op: ”Hoe kan een wetenschappelijk artikel automatisch worden vereenvoudigd, gericht op de unieke noden van scholieren met dyslexie in de 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 voor de taak van gepersonaliseerde en geautomatiseerde tekstvereenvoudiging. De requirementsanalyse wijst uit dat bestaande 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 de derde graad middelbaar onderwijs. Hoewel toepassingen voor gepersonaliseerde automatische tekstvereenvoudiging mogelijk zijn, is het essentieel dat ontwikkelaars zich meer inzetten om de bekende uitdagingen van dyslexie en verwante taalstoornissen bij scholieren weg te werken.

Inhoudsopgave

Lijst van figuren vii

Lijst van tabellen ix

1 Inleiding 1
- 1.1 Probleemstelling . . . 2
- 1.2 Onderzoeksvraag . . . 3
- 1.3 Onderzoeksdoelstelling . . . 4
- 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. . . . 13
  - 2.3.1 Manuele tekstvereenvoudiging . . . 13
  - 2.3.2 Bevoordelende effecten van MTS bij scholieren met dyslexie . . . 14
  - 2.3.3 Aanpak voor ATS. . . . 17
- 2.4 De verschillende soorten ATS . . . 19
- 2.5 Beschikbare tools en taalmodellen . . . 24
- 2.6 De valkuilen bij AI en NLP. . . . 33
- 2.7 Conclusie . . . 35

3 Methodologie 37
- 3.1 Requirementsanalyse . . . 37
- 3.2 Vergelijking van taalmodellen. . . . 43
- 3.3 Prototype voor tekstvereenvoudiging. . . . 51
  - 3.3.1 De ontwikkeling van het scholierencomponent. . . . 69
  - 3.3.2 De opzet voor een lokale webtoepassing. . . . 73
  - 3.3.3 Experimenten met Pentimentor en vergelijkingen met bestaande toepassingen. . . . 74

4 Resultaten 76
- 4.1 Ontbrekende functies bij AI-toepassingen . . . 76

4.2 Geschikte taalmodel voor gepersonaliseerde tekstvereenvoudiging met ATS . . . 82
4.3 Pentimentor vergelijken met *top-of-the-line* tools . . . 89

5 Conclusie **94**

6 Discussie **96**

A Onderzoeksvoorstel **100**
A.1 Introductie . . . 100
A.2 State-of-the-art . . . 102
A.2.1 Tekstvereenvoudiging . . . 102
A.2.2 Noden van scholieren met dyslexie . . . 102
A.2.3 Huidige toepassingen . . . 104
A.2.4 Ontwikkelen met AI . . . 104
A.3 Methodologie . . . 105
A.4 Verwacht resultaat, conclusie . . . 107

B Referentieteksten: Richtlijnen **108**
B.1 Lexicale vereenvoudiging . . . 110
B.2 Syntactische vereenvoudiging . . . 110
B.3 Structurele aanpassingen . . . 110
B.4 Specifieke richtlijnen voor A1 . . . 111
B.5 Specifieke richtlijnen voor A2 . . . 112

Bibliografie **114**

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 benodigde leesgraad voor het lezen van wetenschappelijke artikelen. Bron: (Plavén-Sigray e.a., 2017) . . . 11
2.2 LayoutParser toepassen op drie verschillende documenten. De kaders geven verschillende geclassificeerde tekstblokken aan. Bron: (Shen e.a., 2021). . . . 12
2.3 De gemeten *mean fixation duration* tijdens het begrijpend lezen van teksten uit het onderzoek van Rello, Baeza-Yates, Dempere-Marco e.a. (2013). . . . 15
2.4 Voorbeeld van PoS-labeling uit het artikel van Bilici (2021). . . . 19
2.5 Een pipeline voor LS volgens Althunayyan en Azmi (2021). . . . 20
2.6 Een experiment op de *mean-regret* van GPT-3 engines uit Binz en Schulz (2023). . . . 30
2.7 De werking van *prompt engineering* volgens McFarland (2023) . . . 31
2.8 De evolutie van LLM's. Bron: (Simon, 2021) . . . 33

3.1 Het benodigde stappenplan bij de requirementanalyse. . . . 38
3.2 Het gevolgde stappenplan voor de vergelijking van taalmodellen. . . . 44
3.3 Algemeen overzicht van de ontwikkeling van het prototype voor ATS van wetenschappelijke artikelen. . . . 53

4.1 Informatie opvragen van een wetenschappelijk artikel met SciSpace . 78
4.2 Schermafbeelding van de tekstanalyse bij Simplish na een tekstvereenvoudiging. . . . 79
4.3 Tekstanalyse met *Rewordify*. . . . 80
4.4 Overzicht van het minimum, maximum en gemiddeld aantal woorden per zin per model in A1. . . . 83
4.5 Overzicht van het minimum, maximum en gemiddeld aantal woorden per zin per model in A2. . . . 83
4.6 Boxplot van de FRE-scores voor A1. . . . 84
4.7 Boxplot van de FRE-scores voor A2. . . . 84
4.8 Boxplot van de FOG-scores voor A1. . . . 85
4.9 Boxplot van de FOG-scores voor A2. . . . 85
4.10 Een boxplot van het aantal lange en complexe woorden per zin, gegroepeerd op model voor A1. . . . 86

4.11 Een staafdiagram van het aantal gebruikte hulpwerkwoorden in de tekst, gegroepeerd op model voor A1. 86
4.12 Een staafdiagram van het aantal gebruikte hulpwerkwoorden in de tekst, gegroepeerd op model voor A2. 87
4.13 Het aantal vervoegingen van het werkwoord 'zijn', gegroepeerd op model voor A1. 87
4.14 Het aantal vervoegingen van het werkwoord 'zijn', gegroepeerd op model voor A2. 88
4.15 De verschillen tussen de oorspronkelijke tekst, T4P1 en T4P3 bij één uitgekozen paragraaf. 88
4.16 Voorbeeldweergave van de homepagina. 90
4.17 Voorbeeldweergave van de instellingenpagina. 90
4.18 Voorbeeldweergave van het scholierencomponent. 91
4.19 De uitvoer na een vereenvoudiging met Pentimentor. De tekst is een vereenvoudigde versie van het artikel van Van Brakel (2022). 92
4.20 Een mogelijke weergave van het lerarencomponent met het wetenschappelijk artikel van Van Brakel (2022) als input. 93

A.1 De indeling van leesgraadscores per doelgroep. Bron: (Readable, 2021) 106

Lijst van tabellen

2.1 Unieke drempels bij scholieren met dyslexie tijdens het begrijpend lezen. . . . 7
2.2 Noden en oplossingen om webpagina's beter af te stemmen op de mogelijke noden van scholieren met dyslexie. . . . 8
2.3 Complexe leesfactoren van een wetenschappelijk artikel. . . . 9
2.4 Leesgraadscores volgens onderzoek van Cantos en Almela (2019). . . . 10
2.5 Drie algemene technieken voor MTS bij een algemene doelgroep. . . . 14
2.6 Bewezen voordelen van MTS op mensen met dyslexie tijdens het begrijpend lezen. . . . 17
2.7 Beschikbare Nederlandstalige, Engelstalige en meertalige lexicale databanken anno mei 2023. . . . 21
2.8 Drie manieren om extraherende samenvatting mogelijk te maken volgens Verma en Verma (2020). . . . 23
2.9 Overzicht van gekende voorleessoftware, tekstvereenvoudigings- en samenvattingstools die intuïtief zijn ontwikkeld voor de eindgebruiker (leerkracht of scholier). . . . 26
2.10 Beschikbare en ge-finetunede HF-taalmodellen. . . . 27
2.11 Tabel met alle GPT-3 parameters. . . . 29
2.12 Technieken voor concrete en goed opgestelde prompts. . . . 31
2.13 Samenvattend schema met vaak voorkomende struikelblokken bij NLP-toepassingen. . . . 35

3.1 Shortlist van uit te testen tools en toepassingen voor tekstvereenvoudiging. . . . 38
3.2 Richtlijnen waarop het onderzoek de toepassingen aftoetst in de requirementsanalyse. . . . 39
3.3 Bronvermeldingen voor de twee wetenschappelijke artikelen. . . . 40
3.4 Het moscow-schema voor de requirementsanalyse. . . . 41
3.5 De toegepaste GPT-3-prompts in de requirementsanalyse. . . . 42
3.6 Gebruikte taalmodellen in de vergelijkende studie . . . 43
3.7 Meegegeven parameters bij HF-requests . . . 45
3.8 De GPT-3-prompts die in de vergelijkende studie aan bod komen. . . . 46
3.9 Gebruikte SpaCy word-embeddings . . . 46

3.10 Visualisatietechnieken voor de machinale metrieken bij de vergelijking van de vereenvoudigde teksten met de oorspronkelijke tekst en de referentieteksten. 50
3.11 Criteria voor menselijke observatie bij de vergelijkende studie. 51
3.12 Gebruikte programmeertalen in het prototype voor tekstvereenvoudiging. 52
3.13 Gebruikte Python-libraries en hun respectievelijke functie in het prototype. 54
3.14 Alle beschikbare functionaliteiten in het lerarencomponent. 61
3.15 Tabel met de gebruikte prompts voor het lerarencomponent. 62
3.16 Gebruikte parameters om definities van woorden te genereren met GPT-3. 62
3.17 Benodigde labels voor een gepersonaliseerd document met Pandoc. . 66
3.18 Beschikbare functionaliteiten in het scholierencomponent. 69
3.19 Gekozen parameter voor experimenten. 75

4.1 Afgetoetste criteria volgens de experimenten. 77
4.2 Aantal zinnen (gemeten met Spacy sentence embeddings) per tekst. . 82

List of Listings

3.1 Script voor de eerste fase van de vergelijkende studie. 44
3.2 Script voor de tweede fase van de vergelijkende studie. 44
3.3 Script voor de derde fase van de vergelijkende studie 47
3.4 Script voor de vierde fase van de vergelijkende studie 48
3.5 Code om een boxplot voor het aantal woorden per zin te genereren. . . 49
3.6 De backend functie die de aanpassingen uit het formulier opslaat als sessievariabele. 54
3.7 De onload-functie die de gepersonaliseerde opmaakopties regelt bij het inladen van een webpagina. 55
3.8 Koppeling tussen frontend en backend voor het inlezen van een wetenschappelijk artikel 55
3.9 Een PDF inlezen met PDFMiner 56
3.10 Een PDF inlezen met OCR 57
3.11 Tekst extraheren uit de geparsete inhoud. 58
3.12 Het formatteren van de tekst naar een formaat voor de website. 59
3.13 Het doorlopen van de PDF-tekst op de webpagina en het toekennen van de span-tags. 60
3.14 Implementatie rond localstorage en tekst markeren. 63
3.15 Alle gemarkeerde tekstblokken uit de localstorage aflopen en doorsturen naar het markdown document. 64
3.16 API-calls versturen naar de GPT-3 API. 65
3.17 Writer-klasse omvattende de code om dynamische PDF- en Word-documenten te genereren. 66
3.18 Een woordenlijst genereren met de Writer-klasse. 67
3.19 Een doorlopende tekst toevoegen aan het markdownbestand met de Writer-klasse. 67
3.20 Een opsomming toevoegen aan het markdownbestand met de Writer-klasse. 68
3.21 Een zip-bestand aanmaken met daarin een docx en pdf bestand van de vereenvoudigde tekst. 68
3.22 Een API-call sturen naar GPT-3 met een custom prompt. 70
3.23 Een API-call sturen naar GPT-3 om de definitie van een woord op te halen. 70
3.24 Een woord aan de woordenlijst toevoegen in het scholierencomponent. 71
3.25 Zelfstandige naamwoorden in het scholierencomponent markeren. . . 72

3.26 Dockerfile voor Pentimentor. 73
3.27 Script voor het opstarten van de Docker-container voor Windows-gebruikers 74
3.28 Script voor het opstarten van de Docker-container voor Unix-gebruikers 74

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 leerkrachten divers leesmateriaal gebruiken om lesinhouden op een authentieke manier over te brengen. Zo zetten leerkrachten in de derde graad van het middelbaar onderwijs wetenschappelijke artikelen in als leesvoer. Toch brengt hun leesgraad een nieuwe uitdaging mee voor zowel scholieren als leerkrachten.

Om scholieren attent te maken van wetenschappelijk onderzoek, lanceerde het Amerikaanse onderwijs het C.R.E.A.T.E.[1]-initiatief. Het zet twaalf- tot achttienjarige scholieren aan om wetenschappelijke artikelen te lezen in plaats van enkel boeken. Zo begrijpen ze hoe wetenschappers onderzoek plannen, uitvoeren, en resultaten analyseren en interpreteren. Hoewel er geen vergelijkbare Vlaamse initiatieven bestaan, benadrukken de lerarenopleidingen toch het gebruik van afwisselende leerstof in klas. Andere initiatieven, zoals het M-decreet en de leerplannen van het katholiek[2] en het gemeenschapsonderwijs[3], adviseren Vlaamse leerkrachten om hun lessen op een toegankelijke manier aanbieden. Zo nemen zij alle scholieren ongeacht eventuele leesachterstand mee in het verhaal.

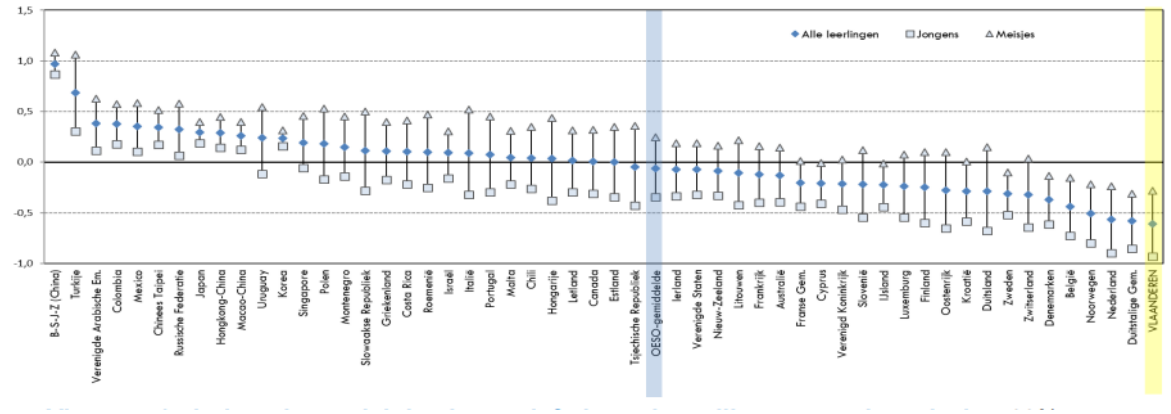
Met een jaarlijks budget van 32 miljoen euro is Vlaanderen een pionier in Europa op het gebied van artificiële intelligentie (AI) op de werkvloer (Crevits, 2022). Zo stampte de Vlaamse overheid verschillende projecten uit de grond om Vlaamse AI ontwikkelingen te ondersteunen en AI softwarebedrijven te inspireren. Het amai!-project[4] brengt AI softwarebedrijven uit diverse domeinen samen, waaronder het

[1]https://teachcreate.org/
[2]https://pro.katholiekonderwijs.vlaanderen/basisoptie-stem/ondersteunend-materiaal
[3]https://g-o.be/stem/
[4]https://amai.vlaanderen/

onderwijs. Zij doelen op een automatisering van processen om de werkdruk bij leerkrachten te verminderen. Dit gebeurt onder andere door *real-time* ondertiteling in de klas en een taalassistent voor leerkrachten in meertalige klasgroepen.

1.1. Probleemstelling

De geletterdheid van scholieren bevindt zich op een kritieke punt. Elke drie jaar nemen experts uit 79 geïndustrialiseerde landen de PISA-test af bij middelbare scholen 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 dit cijfer hoger dan 20%. Lezen kan daarmee een obstakel vormen 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 Onderwijsinspectie Overheid Vlaanderen (2020) dat begrijpend lezen een essentiële vaardigheid is, 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. Ondanks de bestaande ondersteuning blijven ze toch nog steeds de negatieve impact van hun leerstoornis ervaren. De gevolgen hiervan kunnen zich doorzetten na het mid-

delbaar 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 probleemoplossers zijn (Bonte, 2020; Ghesquière, 2018; Lissens e.a., 2020).

Het leerplan voor STEM-vakken stimuleert het gebruik van wetenschappelijke artikelen, maar houdt niet altijd rekening met de bijhorende en complexe leesgraad. De ingewikkelde woordenschat en syntax in wetenschappelijke artikelen kunnen een hindernis vormen voor de begrijpelijkheid van een tekst, aldus Plavén-Sigray e.a. (2017). Wetenschappelijke artikelen handmatig vereenvoudigen kan planning, tijd en energie van leerkrachten in het middelbaar onderwijs opslorpen. Het Vlaamse onderwijssysteem 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 te bieden aan het middelbaar onderwijs (Belpaeme e.a., 2018). Het onderwijst past echter zelden soortgelijke technologieën toe. Er is terughoudendheid door enerzijds ouders van leerlingen volgens 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 (Kraft, 2020). 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 de werkdruk bij leerkrachten verminderen, en anderzijds scholieren in de derde graad ondersteunen bij het lezen van complexe wetenschappelijke artikelen.

1.2. Onderzoeksvraag

Dit onderzoek beschrijft het gebruik van AI 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 stelt het onderzoek de volgende onderzoeksvraag op:

- 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 voor de volgende deelvragen vormen een globaal antwoord 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 intuïtieve lokale webtoepassing die zowel scholieren met dyslexie als leerkrachten helpt?

1.3. Onderzoeksdoelstelling

Het onderzoek achterhaalt de technologische en logopedische aspecten waarmee ontwikkelaars rekening meoten houden bij AI-tekstvereenvoudiging. Het resultaat dient als een houvast om hen te begeleiden tijdens de ontwikkeling van deze applicaties voor gepersonaliseerde en geautomatiseerde tekstvereenvoudiging. Verder ontwikkelt het onderzoek een soortgelijke toepassing in het bijzonder voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Dit resulteert in een uitgewerkt prototype voor tekstvereenvoudiging, genaamd *Pentimentor*. *Pentimentor* heeft voornamelijk twee functies. Allereerst kan *Pentimentor* wetenschappelijke artikelen vereenvoudigen op basis van de specifieke behoeften van scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

Daarnaast biedt *Pentimentor* een geautomatiseerde benadering om wetenschappelijke artikelen op een gepersonaliseerde manier te vereenvoudigen door het gebruik van aanpasbare parameters. Tot slot geeft *Pentimentor* de eindgebruiker het vereenvoudigd artikel terug in Word-formaat.

1.4. Opzet van deze bachelorproef

De rest van deze scriptie is als volgt opgebouwd:

- Hoofdstuk 2 geeft een overzicht van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.
- Hoofdstuk 3 licht de methodologie toe. Het onderzoek vermeldt de gebruikte onderzoekstechnieken om een antwoord te kunnen formuleren op de onderzoeksdeelvragen.
- Hoofdstuk 4 bevat de resultaten voor alle onderzoekstechnieken.
- Hoofdstuk 5 geeft de uiteindelijke conclusie en beantwoordt daarmee de onderzoeksvraag.
- Tot slot 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 benodigde kennis binnen het logopedisch, taal- en informaticavakdomein 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 taalexperten op verschillende manieren. Het is belangrijk om stil te staan bij de bestaande en reeds bewezen technieken voor *manual text simplification*. Vervolgens komen technieken voor *automated text simplification* 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 specifiek voor scholieren met dyslexie te vereenvoudigen, moet het onderzoek stilstaan bij de unieke noden van scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Daarnaast moet het onderzoek stilstaan bij de moeilijkheden tijdens het begrijpend lezen van wetenschappelijke artikelen. Deze sectie bespreekt eerst welke technieken en metho-

den er bestaan om scholieren met dyslexie te ondersteunen tijdens het begrijpend lezen van teksten. Daarna worden de belemmeringen en moeilijkheden van wetenschappelijke artikelen aangekaart. Deze sectie beantwoordt de volgende twee onderzoeksvragen:

- 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. Tabel 2.1 somt deze noden op.

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 lettergreepcombinaties	Gala en Ziegler (2016)

Tabel 2.1: Unieke drempels bij scholieren met dyslexie tijdens het begrijpend lezen.

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 op (Botelho, 2021). Verder omschrijft 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 toont deze noden.

Kenmerk	Bron
Zachtgele, -groene -of bruine achtergrondkleur	Rello en Bigham (2017) en Santana e.a. (2012)
Grotere lettergrootte dan 14pt gebruiken	(Rello & A. Baeza-Yates, 2015)
Woord- en karakterspatiëring verbreden	Rello en Baeza-Yates (2013) en Santana e.a. (2012)
Consistente lay-out toepassen	(Botelho, 2021; Rello & A. Baeza-Yates, 2015)
Waarschuwingen geven omtrent formulieren, sessies (login)	(Botelho, 2021)
Duidelijk zichtbare koppen- of headingstructuur	(Rello e.a., 2012a)
Duidelijke symbolen of *icons* gebruiken	(Rello e.a., 2012b)
Inhoud visueel groeperen	(Botelho, 2021; Rello & A. Baeza-Yates, 2015)
Huidige positie benadrukken	(Botelho, 2021)

Tabel 2.2: Noden en oplossingen om webpagina's beter af te stemmen op de mogelijke noden van scholieren met dyslexie.

2.2.2. Specifieke kenmerken van wetenschappelijke artikelen

Wetenschappelijke artikelen zijn van cruciaal belang voor het verspreiden van nieuwe kennis en onderzoeksresultaten. Toch blijven ze voor velen een mysterieus en ontoegankelijk gebied, omwille van 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. Daarmee vormt er zich een extra obstakel bij het implementeren van wetenschappelijke artikelen als bron van kennis tijdens de les. Zo volgen wetenschappelijke artikelen 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 en leren ze een kritische vaardigheid.

Probleem	**Oplossing**	Bron
Veel informatie in een compact formaat of *high information density*	Extra uitleg schrijven bij woorden of compacte zinnen schrijven.	(Matarese, 2013; Plavén-Sigray e.a., 2017)
Hoog gebruik van meerlettergrepige woorden	Synoniemen met minder lettergrepen gebruiken.	(Siddharthan, 2006)
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.	(Plavén-Sigray e.a., 2017)
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.	(Pain, 2016)
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'.	(Plavén-Sigray e.a., 2017)

Tabel 2.3: Complexe leesfactoren van een wetenschappelijk artikel.

Scholieren kunnen over verschillende achtergrondkennis beschikken, wat invloed kan hebben op het tekstbegrip tijdens het begrijpend lezen (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 zorgen voor de ontwikkeling van geautomatiseerde berekening van leesmetrieken en leesgraadsscores. Zo kunnen ontwikkelaars dit doen met python-libraries via commandline of CLI-toepassingen, waaronder Text-

stat[1] en Readability[2]. Tabel 2.4 toont drie prevalente leesgraadscores weer. Daarnaast kunnen toepassingen, zoals Textinspector[3] of Charactercalculator[4], analytisch inzicht geven in de complexiteit van Engelstalige teksten. Deze toepassingen kunnen echter geen Nederlandstalige teksten of leesmetrieken analyseren. Hoewel deze leesmetrieken een beknopte analyse kunnen vormen voor taaldeskundigen, toch benadrukken onderzoekers dat deze leesgraadscores geen rekening houden met de achtergrondkennis van mogelijke lezers, aldus (Cantos & Almela, 2019). Recent onderzoek van Crossley e.a. (2019) achterhaalt de mogelijkheid om geautomatiseerde taalverwerking te gebruiken voor leesmetrieken die wel rekening houden met de doelgroep. Hoewel deze modellen potentieel tonen, kunnen ontwikkelaars deze nog niet gebruiken (Crossley e.a., 2019).

Score	**Uitleg**
Flesch Reading Ease (FRE)	Deze leesgraadscore berekent de moeilijkheidsgraad op zinbasis. Hoe hoger de score, hoe 'eenvoudiger' de zin (Cantos & Almela, 2019; Readable, 2021).
Gunning FOG (FOG)	In tegenstelling tot FRE, berekent FOG de moeilijkheidsgraad op basis van de volledige tekst (Cantos & Almela, 2019).
Complexe woordenlijst volgens *Dale-Chall Index* (DCI)	Deze lijst omvat woorden die experimenten bij Amerikaanse tieners als complex omschrijven. De DCI werkt per leeftijdscategorie (Cantos & Almela, 2019).

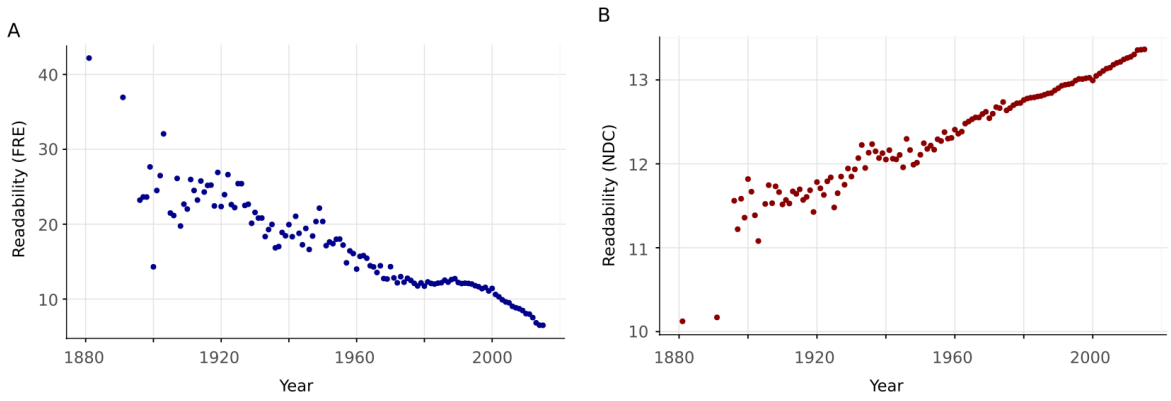
Tabel 2.4: Leesgraadscores volgens onderzoek van Cantos en Almela (2019).

Divers onderzoek 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) maken wetenschappers en onderzoekers onbewust wetenschappelijke artikelen moeilijker om te lezen. Uit een vergelijkende studie tussen abstracten en de rest van de inhoud van wetenschappelijke tijdschriften blijkt dat abstracts het meest complexe deel van een artikel vor-

[1] https://pypi.org/project/textstat/
[2] https://pypi.org/project/readability/
[3] https://textinspector.com/
[4] https://charactercalculator.com/

men. De evolutie van de leesbaarheid wordt weergegeven in figuur 2.1. Deze figuur toont de FRE (links) en NDC (rechts) scores. Zo schat het onderzoek 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.



Figuur (2.1)
De toename van benodigde leesgraad voor het lezen van wetenschappelijke artikelen. 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 zijn de onderzoeken van Hartley (1999) en Snow (2010) relevant waarin ze aantonen dat het herschrijven van abstracts de begrijpbaarheid ervan kan verhogen.

Volgens Hollenkamp (2020) 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, zoals PDFMiner of PyPDF. Wel ondervinden ontwikkelaars soms problemen bij het inlezen van pdf-bestanden,

aldus Lee (2021). Tools kunnen niet alle tekstinhoud uit een pdf extraheren. Als oplossing kunnen ontwikkelaars gebruik maken van *optical character recognition* of OCR. Ondertussen bestaan er python-bibliotheken die deze technologie met een eenvoudige implementatie kunnen uitwerken, namelijk EasyOCR[5] en Tesserat (Lee, 2021).

Bovendien bestaat de uitvoer van deze artikelen uit louter losse tekstblokken. Het systeem is niet in staat om automatisch te identificeren welke delen titels, afbeeldingen of tekstblokken zijn. Een mogelijke oplossing hiervoor is het gebruik van *LayoutParser*[6]. Dit is een *deep learning* of DL-model dat zorgvuldige *Document Image Analysis* uitvoert. Met behulp van *LayoutParser*, in samenwerking met het Detectron2[7]-algoritme, is het mogelijk om de tekstinhoud van wetenschappelijke artikelen te extraheren. Figuur 2.2 geeft een voorbeeld weer waarbij het model de tekstblokken van drie verschillende soorten documenten, waaronder een wetenschappelijk artikel links in beeld, markeert en classificeert. Met reeds vermelde OCR-technieken kan het systeem deze omkaderde tekst inlezen en gebruiken in dergelijke toepassingen (Shen e.a., 2021).

Figuur (2.2)
LayoutParser toepassen op drie verschillende documenten. De kaders geven verschillende geclassificeerde tekstblokken aan. Bron: (Shen e.a., 2021).

Conclusie

In deze eerste sectie van de literatuurstudie zocht het onderzoek naar de specifieke behoeften van scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Verder stond het stil bij de moeilijkheden die zij ervaren bij het begrijpend lezen van wetenschappelijke artikelen. Zo wijst de literatuurstudie de volgende technieken en methoden om hen te ondersteunen bij het begrijpend lezen. Tabellen 2.2 en 2.1 geven een overzicht van deze technieken en methoden. Daarnaast

[5]https://pypi.org/project/easyocr/
[6]https://pypi.org/project/layoutparser/
[7]https://ai.meta.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/

omschrijven onderzoeken ook specifieke kenmerken van wetenschappelijke artikelen die het begrip ervan bemoeilijken, opgesomd in tabel 2.3. Tot slot kunnen ontwikkelaars en vakexperten de complexiteit van een tekst berekenen met bestaande leesgraadscores, zoals beschreven in tabel 2.4.

2.3. Aanpakken voor tekstvereenvoudiging

2.3.1. Manuele tekstvereenvoudiging

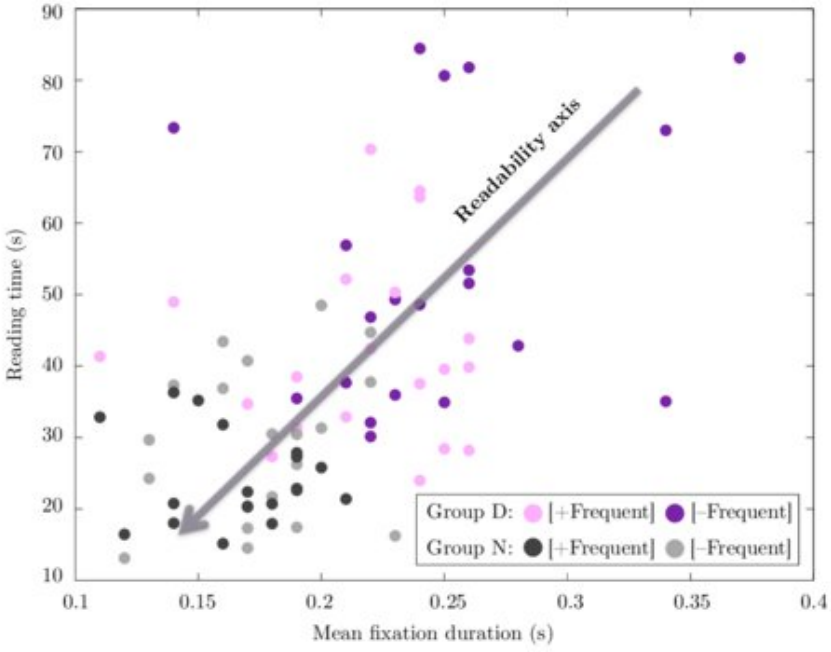
Voor sommige lezers kan het begrijpend lezen van een complexe tekst echter een uitdaging zijn, zoals scholieren met dyslexie. *Manual tekst simplification* of MTS kan deze groep helpen (Siddharthan, 2014). De techniek van MTS gebruikt eenvoudige woordenschat en zinsstructuren en maakt structurele aanpassingen (SA) om de tekst vlotter leesbaar te maken. MTS is het proces dat het technische leesniveau en het woordgebruik van een geschreven tekst vermindert. Dit resulteert tot een betere leeservaring zonder het verlies van de kerninhoud tijdens het lezen van de tekst. Tabel 2.5 toont een overzicht van bewezen MTS-technieken, zonder een specifiek toespitsing op een doelgroep.

Type vereen-voudiging	Techniek	Bron
LS	Moeilijke woorden vervangen door eenvoudigere synoniemen	(Crossley e.a., 2012; Rello, Baeza-Yates & Saggion, 2013; Siddharthan, 2014)
	Woorden- en synoniemenlijst maken	(Bosmans e.a., 2022b; Siddharthan, 2006)
	Dubbelzinnige woorden vervangen	
	Idiomen vervangen	(Siddharthan, 2006)
	Regelmatige lettergreepcombinaties gebruiken	(Gala & Ziegler, 2016)
	Rekening houden met het gekende jargon van de doelgroep	(Javourey-Drevet e.a., 2022)
SS	Tangconstructies aanpassen	(Bosmans e.a., 2022a)
	Zinnen langer dan tien woorden inkorten	(Siddharthan, 2014)
	Verwijswoorden aanpassen	(Bosmans e.a., 2022c)
	Voorzetseluitdrukkingen aanpassen	(Rello, Baeza-Yates, Bott e.a., 2013)
	Samengestelde werkwoorden aanpassen	(Bosmans e.a., 2022b)
	Actieve stem gebruiken	(Ruelas Inzunza, 2020)
	Onregelmatige werkwoorden vermijden	(Gala & Ziegler, 2016; Rello, Baeza-Yates, Bott e.a., 2013)
SA	Marges aanpassen	(Rello, Baeza-Yates, Bott e.a., 2013)
	Lettertype -en grootte aanpassen	(Rello e.a., 2012a)
	Woord- en karakterspatiëring aanpassen	(Rello e.a., 2012a)
	Herschrijven als opsomming of tabelvorm	(Rello & A. Baeza-Yates, 2015)

Tabel 2.5: Drie algemene technieken voor MTS bij een algemene doelgroep.

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



Figuur (2.3)
De gemeten *mean fixation duration* tijdens het begrijpend lezen van teksten uit het onderzoek van Rello, Baeza-Yates, Dempere-Marco e.a. (2013).

e.a., 2021). Bovendien blijkt uit experimenten dat frequent woordgebruik de decodeertijd bij mensen met dyslexie significant vermindert, en dat teksten met verminderde lexicale complexiteit minder leesfouten opleveren voor mensen met dyslexie (Gala & Ziegler, 2016; Rello, Baeza-Yates, Dempere-Marco e.a., 2013). De studie van Gala en Ziegler (2016) benadrukt 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 2.3. Het gebruik van veelvoorkomende woorden vermindert de decodeertijd en verbetert het leesbegrip voor mensen met dyslexie.

Hoewel onderzoeken de positieve effecten van lexicale vereenvouding 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. Het onderzoek van Leroy e.a. (2013) onderzoek het effect van herstelde coherentieonderbrekingen en plaatste tekst in een logische volgorde. Zo 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 tijdens het onderzoek 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 andere 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 rusulterende 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 (Rello & Baeza-Yates, 2013). Volgens Bezem en Lugthart (2016), Rello en A. 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 A. Baeza-Yates (2015) en Rello en Baeza-Yates (2013).

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	(Gala & Ziegler, 2016; Rello, Baeza-Yates, Dempere-Marco e.a., 2013)
	Beter leesbegrip	(Gala & Ziegler, 2016; Rello, Baeza-Yates, Dempere-Marco e.a., 2013)
Verwerpen van onregelmatige lettergrepen	Verminderde decodeertijd	(Gala & Ziegler, 2016)
	Beter leesbegrip	(Gala & Ziegler, 2016)
Causale structuren aanpassen	Beter leesbegrip	(Linderholm e.a., 2000)
	Minder fouten bij het begrijpend lezen	(Linderholm e.a., 2000)
Tekstgebeurtenissen in een tijdsafhankelijke volgorde plaatsen	Beter leesbegrip bij het reviseren	(Leroy e.a., 2013)
Coherentieonderbrekingen herstellen	Beter leesbegrip bij het reviseren	(Leroy e.a., 2013)
Gepersonaliseerde samenvatting	Betere leesbaarheid	(Nandhini & Balasundaram, 2013)
Zachtkleurige achtergrond	Betere leesbaarheid	(Rello & A. Baeza-Yates, 2015)
Niet-cursieve, sans-serif lettertypen	Betere leesbaarheid	(Rello & Baeza-Yates, 2013)
Lettertype groter dan 14pt	Betere leesbaarheid	(Rello & Baeza-Yates, 2013)

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 -en 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 daaropvol-

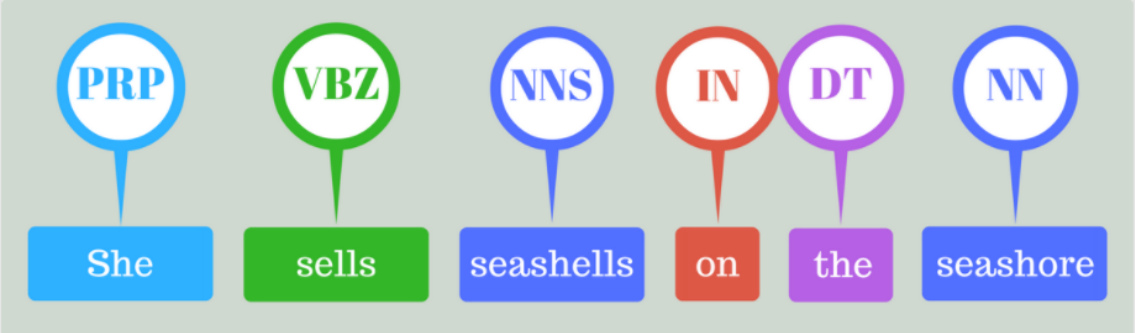
gende concepten.

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 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 JohnSnow[8]. 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 2.4 en is afkomstig uit Bilici (2021).

[8]https://nlp.johnsnowlabs.com/2020/05/03/lemma_nl.html



Figuur (2.4)
Voorbeeld van PoS-labeling uit het artikel van Bilici (2021).

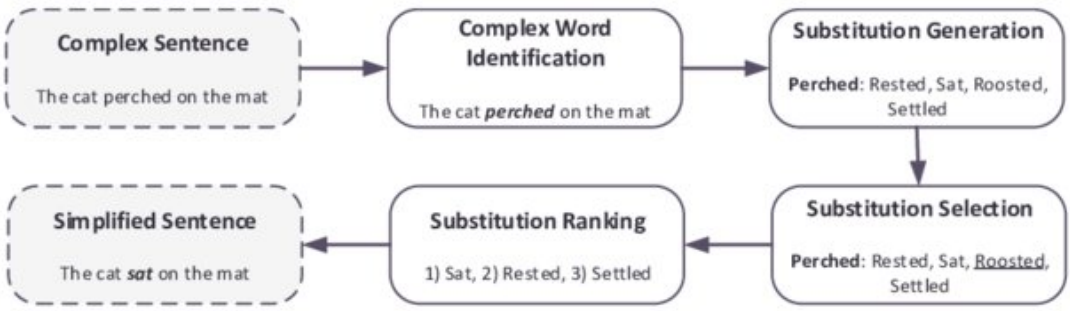
Met NER-labeling kunnen systemen zo namen van personen, organisaties en locaties herkennen en classificeren. Het haalt specifieke informatie uit een tekst, zoals het identificeren van de namen van personen, plaatsen of bedrijven die in nieuwsartikelen, of het extraheren van belangrijke data of getallen 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) hebben een neuraal 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 wel de context van een woord (Eisenstein, 2019).

2.4. De verschillende soorten ATS

Tekstvereenvoudiging kan bijdragen aan het begrijpen van complexe informatie. Zoals onderzocht door Siddharthan (2014), zijn er vier soorten transformaties bij geautomatiseerde tekstvereenvoudiging, 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 2.5 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. Recenter 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 (2.5)
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.

Syntactic simplification (SS) of syntactische vereenvoudiging is een techniek 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 semantiek bewaren blijft. Kandula e.a. (2010) onderzochten de ontwikkeling van dergelijk model 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 OpenNLP[9] 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.

ATS is geen nieuw concept. Volgens onderzoeken van Canning e.a. (2000) en Siddharthan (2006) zijn de eerste aanpakken op ATS gebouwd op rule-based modellen. Deze modellen bewerken de syntax door zinnen te splitsen, te verwijderen of de volgorde van de zinnen in een tekst aan te passen. LS komt hier niet aan de pas. Recentere onderzoeken van Bulté e.a. (2018) en Coster en Kauchak (2011) verduidelijken hoe ontwikkelaars LS en SS kunnen combineren.

[9]https://opennlp.apache.org/

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 LS en SS 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). Tabel 2.8 omschrijft deze drie methoden verder.

MMR-gebaseerde ES	Deze techniek gebruikt de maximaal marginale relevantie-score (MMR) om de relevantie en diversiteit van gemarkeerde zinnen te bepalen. Dit voorkomt dat geselecteerde zinnen elkaar niet overlappen in inhoud en relevantie. Deze methode kan leiden tot betere samenvattingen, maar vereist meer rekenkracht en tijd dan de andere twee technieken.
Graafgebaseerde ES	Deze techniek vertegenwoordigt een document als een graaf van zinnen. Deze vorm gebruikt algoritmen om de belangrijkste zinnen te bepalen en redundantie te vermijden. Dit 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 om de belangrijkste zinnen in een tekst te vinden (Premjith e.a., 2015; Verma & Verma, 2020). Evaluatiefuncties kunnen echter afhankelijk van de gebruikte criteria in een lokaal optimum vastlopen (Rani & Kaur, 2021).

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

Vooroordelen of *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) bouwden verder op deze experimenten door het combineren van *knowledge-rich* en *knowledge-poor* methoden, wat resulteerde 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 *hiërarchical attention network*.

Extraherende samenvattingen kunnen leiden tot een onsamenhangende 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 samengevatte 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 samenhangende 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.

Tot slot moeten ontwikkelaars 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 geautomatiseerde 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; Departement onderwijs en vorming, 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 beschikken de voorleessoftware beperkte LS-functionaliteiten. Dit onderstreept de noodzaak aan nieuwe erkende tools die tekstvereenvoudiging in het onderwijs mogelijk maken. Tools zoals Simplish en Rewordify kunnen een oplossing bieden. Hoewel Simplish oorspronkelijk Engelstalig is, kan het inmiddels vereenvoudigde teksten genereren van Nederlandstalige teksten. Deze functionaliteit is echter enkel betalend voor niet-Engelstalige teksten. Vervolgens kan Rewordify enkel Engelstalige teksten vereenvoudigen. Tot slot vindt de literatuurstudie weinig online *proof-of-concepts* terug. Daarnaast bieden de toepassingen geen transparantie over hun gebruikte taalmodel, waardoor ontwikkelaars de logica en werking van deze toepassingen niet kunnen reproduceren.

Voor samenvatting zijn er echter meer tools beschikbaar. Enkele voorbeelden hiervan zijn Resoomer, Paraphraser, Editpad, Scribbr en Quillbot. Al zijn er onderzoeken over ATS-technieken 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 SciSpace[10] en Scholarcy[11] ATS specifiek voor wetenschappelijke artikelen aan. 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 prevalente tools die momenteel tekstvereenvoudiging aanbieden.

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 tekstvereenvoudiging 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 (HF) een breed scala aan API's en tools die gemakkelijk te downloaden en trainen zijn

[10] https://typeset.io/
[11] https://www.scholarcy.com/

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. Tot slot geeft tabel 2.10 HF-taalmodellen met hun respectievelijke casus. De volgende taalmodellen neemt de tabel op: Google Pegasus[12], Longformer Encoder-Decoder[13], Simplification[14], BART Large Scientific Summarisation[15], T5 finetuned text simplification model[16] en Keep It Simple[17]

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

Tabel 2.10: Beschikbare en ge-finetunede HF-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 modelleing goal*. Ontwikkelaars trainden dit model op niet-gecategoriseerde data van het internet met datasets zoals Common Crawl, WebText2, Books1, Books2 en Wikipedia.

[12]https://ai.googleblog.com/2020/06/pegasus-state-of-art-model-for.html
[13]https://huggingface.co/docs/transformers/model_doc/led
[14]https://huggingface.co/haining/scientific_abstract_simplification
[15]https://huggingface.co/sambydlo/bart-large-scientific-lay-summarisation
[16]https://huggingface.co/husseinMoh/t5-small-finetuned-text-simplification
[17]https://huggingface.co/philippelaban/keep_it_simple

- Tenslotte finetunen de ontwikkelaars dit verder. Om een correcte respons van het model te krijgen, passen de ontwikkelaars *reinforcement learning* toe.

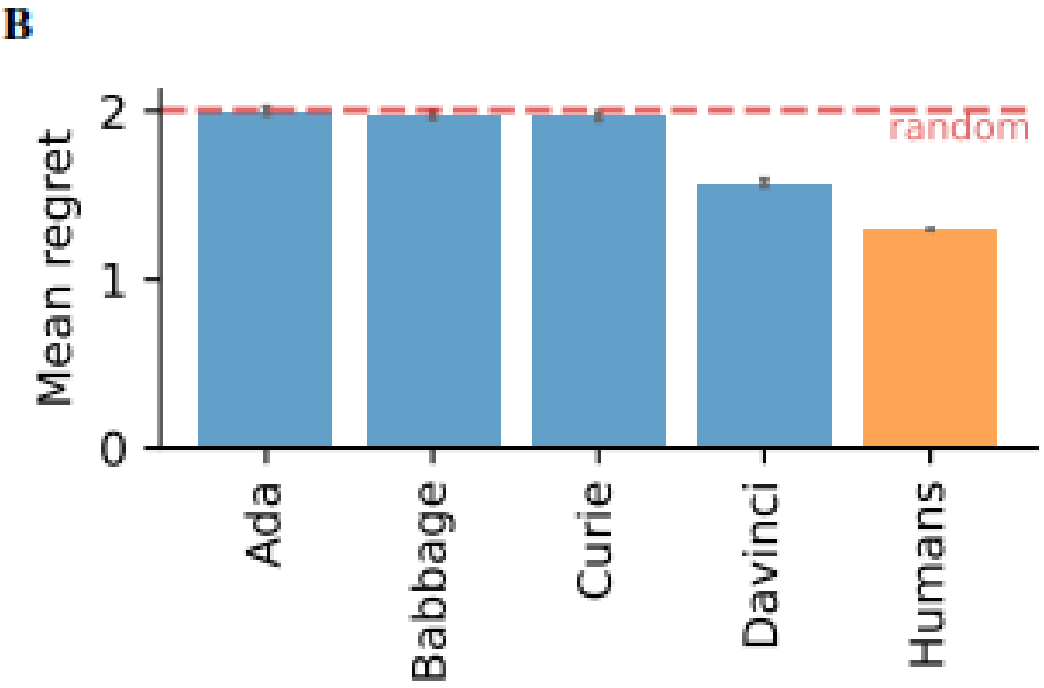
GPT-3 beschikt over meerdere versies, waaronder GPT-3.5 die als engine dient voor ChatGPT. Omdat onbegrijpelijke en ontoegankelijke zinsstructuren niet alleen voor leken, maar ook voor vakexperten een obstakel vormen, is het belangrijk om te benadrukken dat GPT-3.5 gericht is op conversationele doeleinden, terwijl GPT-3 in het algemeen bedoeld is om met hoogstens één prompt te werken (Hubbard & Dunbar, 2017; McNutt, 2014). Verder reikt OpenAI documentatie uit 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 voorspelbare 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 *probability distribution* voor *common tokens*. 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 2.6. Uit het experiment van Tanya Goyal (2022) blijkt dat *zero-shot* samenvattingen met GPT-3 beter presteren dan gefinetunede modellen. C. Li

(2022) benadrukt dat GPT-3 overkill is voor sentimentanalyse. Daarvoor haalt het onderzoek aan om een kleinschaliger taalmodel te gebruiken. Daarnaast beschikken LLM's over een grotere ecologische voetafdruk, waarvoor onderzoeken van Simon (2021) en Strubell e.a. (2019) deze praktijk ook afraden. Uiteindelijk 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).



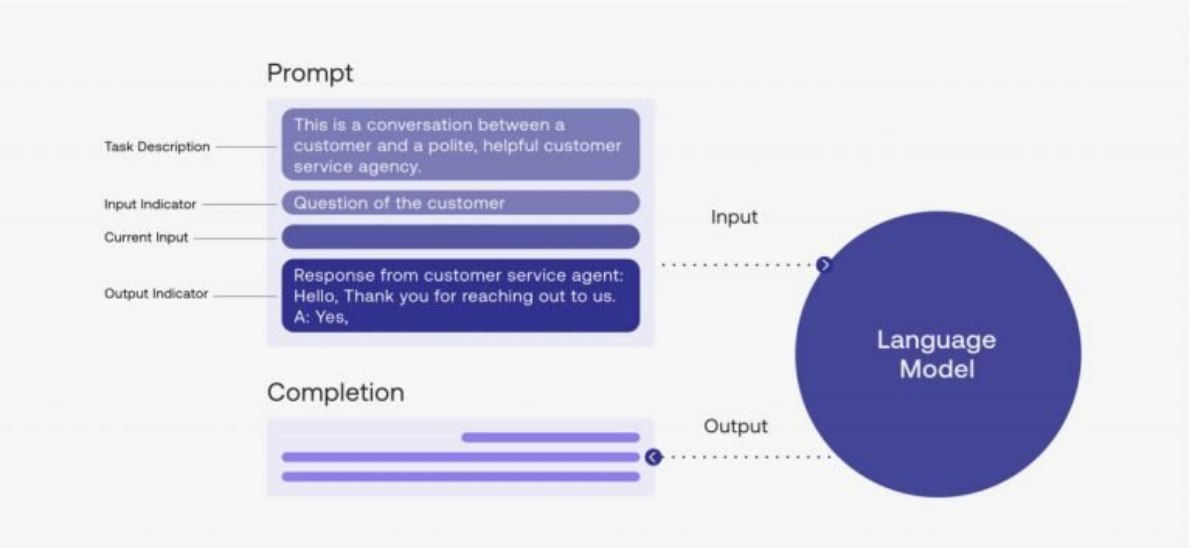
Figuur (2.6)
Een experiment op de *mean-regret* van GPT-3 engines uit Binz en Schulz (2023).

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 2.7 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 deze werk kunnen produceren op maat van het doel. Zo benadrukt de onderzoe-

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.

ker 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 (2.7)
De werking van *prompt engineering* volgens McFarland (2023)

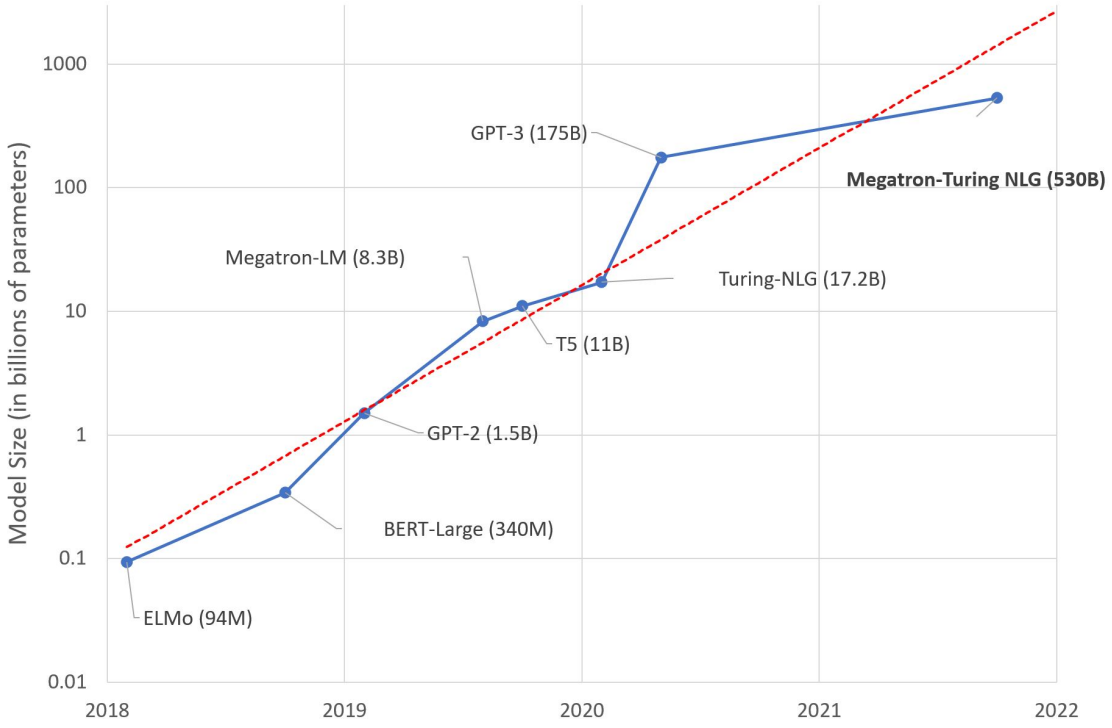
Om prompts te kunnen hergebruiken, bouwden ontwikkelaars *prompt patterns* op. Deze patronen bieden herbruikbare oplossingen voor veelvoorkomende problemen in een bepaalde context, vooral bij de interactie met LLM's. White e.a. (2023) benoemt vier *prompt patterns*:

- *Intent prompts* waarbij een LLM een instructie krijgt met een specfiek 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 die 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 zincontext noodzakelijk is. Omdat GPT-3 toegang heeft tot meer informatie (45 TB) dan BERT (3 TB), kan dit een voordeel bieden tijdens taalbewerkingen zoals vereenvoudigen, parafraseren of vertalen. Tot slot 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 2.8 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 (2.8)
De evolutie van LLM's. Bron: (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 CLI-toepassingen 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 genegligeerde uitdagingen en valkuilen (Khurana e.a.,

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

- 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 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).

Homoniemen kunnen *sequence labeling* bemoeilijken, 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-systemen met synoniemen en antoniemen kan worden bereikt door het gebruik van *candidate generation* en synoniemherkenning, terwijl meertalige transformers zoals BERT een oplossing bieden voor de beperkte toepassingen in andere talen dan het Engels (Dandekar, 2016; Roldós, 2020).

Verschillende studies tonen aan dat MTS en ATS gelijke kansen kunnen bieden aan iedereen. Bij de ontwikkeling van gepersonaliseerde ATS-toepassingen moeten ontwikkelaars de ethische overwegingen en de behoeften van de eindgebruiker in overweging nemen, zoals beschreven in onderzoeken van Gooding (2022), Niemeijer e.a. (2010) en Xu e.a. (2015). Om de eindgebruiker meer controle te geven, moet deze eindgebruiker kunnen kiezen welke delen van de tekst het systeem moet vereenvoudigen.

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 achting nemen.

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

2.7. Conclusie

Tot slot beantwoordt de literatuurstudie de onderzoeksdeelvragen. Zo is begrijpend lezen voor scholieren met dyslexie in de derde graad van het middelbaar niet enkel moeizaam, maar gaat verder dan dat. Deze scholieren kunnen moeite onder-

vinden bij het ontcijferen en automatiseren van woordherkenning. MTS en aangepaste opmaakopties bieden bewezen voordelen voor scholieren met dyslexie. 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. Zo kunnen MTS-technieken, besproken in 2.6, scholieren met dyslexie helpen bij het begrijpend lezen van wetenschappelijke artikelen. Tabel 2.3 somt alle complexe leesfactoren op, alsook hoe lezers deze moeilijke materie kunnen aanpakken. Deze twee tabellen moeten dienen als aftoetscriteria bij de ontwikkeling van een prototype voor gepersonaliseerde 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. Zo kunnen de achterliggende taalmodellen van ChatGPT en Bing Chat specifieke vragen verwerken. Het onderzoek moet verschillende prompts kunnen vergelijken bij het uittesten van dit taalmodel. Verschillende LS, SS en SA-technieken moeten in deze vergelijkende studie aan bod komen.

Gerichte menselijke vergelijkingen en leesbaarheidsformules dienen als evaluatie voor de vereenvoudigde teksten door ATS. De python-bibliotheek *readability* biedt de leesgraadscores, zoals weergegeven in 2.4, aan voor ontwikkelaars. Zo kan dit onderzoek de uitvoer van verschillende teksten vergelijken met een objectieve maatstaf.

Tot slot geeft tabel 2.13 een overzicht van struikelblokken waarmee ontwikkelaars rekening moeten houden. In het kader van ATS voor wetenschappelijke artikelen, moet het taalmodel gebruikmaken of voldoende getraind zijn op data van wetenschappelijke artikelen en vereenvoudigde versies van diezelfde wetenschappelijke artikelen. Als het onderzoek gebruik maakt van een promptgebaseerd model, dan moeten deze prompts op maat gemaakt zijn voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Tot slot moet dergelijk prototype ook de gebruiker de vrijheid krijgen om teksten te markeren waarvan deze persoon een vereenvoudigde versie wilt.

3

Methodologie

De vergaarde kennis uit de literatuurstudie dient als vertrekpunt voor het verdere onderzoek. Zo kan het de onderzoeksvraag beantwoorden met drie onderzoeksstappen. Eerst staat het onderzoek stil bij de nodige functionaliteiten om gepersonaliseerde ATS te kunnen verwezenlijken. Vervolgens achterhaalt het een geschikt taalmodel voor gepersonaliseerde ATS. Tot slot ontwikkelt het onderzoek Pentimentor, ofwel het prototype voor ATS van wetenschappelijke artikelen. Dit moet een eenduidige manier voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs aanreiken om wetenschappelijke artikelen te vereenvoudigen. Zo doelt dit onderzoek om de haalbaarheid voor een toepassing voor gepersonaliseerde ATS te achterhalen met Pentimentor.

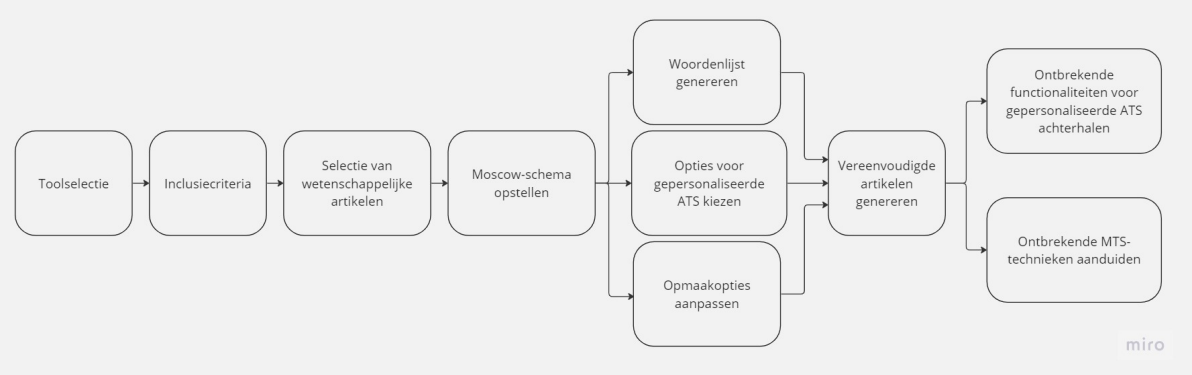
3.1. Requirementsanalyse

Om het ontwikkelingsproces van Pentimentor gericht te sturen, moet het onderzoek bestaande en haalbare 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 en haalbare functionaliteiten voor een ATS-toepassing definieert. Met dit kan het onderzoek een vergelijkbare toepassing ontwikkelen voor gepersonaliseerde ATS van wetenschappelijke artikelen met de kwaliteiten van gepersonaliseerde MTS. Daarnaast achterhaalt deze fase 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 onderzoeksmethode kan het onderzoek een antwoord geven op de volgende twee deelvragen van het onder-

zoek.

- 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 LS toepassen. Buiten 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 betrekt 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)	Simplish (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 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 als een inschatting van de functionaliteiten van deze tools.

MTS-techniek	**Functionaliteit**
LS	Gepersonaliseerde LS, ofwel woordenschat dat niet te hooggegrepen 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
SS	Zinnen inkorten Verwijswoorden aanpassen Voorzetseluitdrukkingen aanpassen Samengestelde werkwoorden aanpassen Actieve stem toepassen Enkel regelmatige werkwoorden gebruiken
SA	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 aftoetst 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 van de functionaliteiten volgens prioriteit te verkrijgen, bouwt het onderzoek een moscow-schema op vanuit de opgestelde richtlijnen. Zo komen belangrijke functionaliteiten, die nodig zijn om gepersonaliseerde tekstvereenvoudiging met ATS mogelijk te maken, in de categorie *must-haves* terecht. Daarnaast omvatten *should-haves* alle functionaliteiten die niet bijdragen tot gepersonaliseerde ATS, maar wel een meerwaarde biedt in dergelijke toepassingen. Vervolgens bestaan de *could-haves* uit alle functionaliteiten die huidige toepassingen kunnen doen, maar weinig bijdragen tot gepersonaliseerde ATS. Irrelevante functionaliteiten binnen de scope van een prototype of niet toepasselijk voor de doelgroep plaatst het onderzoek als *wont-have*.

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

Tabel 3.4: Het moscow-schema voor de requirementsanalyse.

Vervolgens komen experimenten op functionaliteiten voor gepersonaliseerde ATS op wetenschappelijke artikelen aan bod. Allereerst moeten eindgebruikers wetenschappelijke artikelen kunnen opladen. Indien het onderzoek een wetenschappelijk artikel als pdf kan opladen, dan extraheert het de tekstinhoud van het wetenschappelijk artikel. Vervolgens krijgt de toepassing deze tekst als invoer. Zo krijgen de chatbots eerst de prompt, gevolgd door een stuk van het wetenschappelijk artikel. Met zes verschillende prompts kan het onderzoek de LS, SS en SA-functionaliteiten van een promptgebaseerde toepassing achterhalen. Tabel 3.5 vermeldt de toegepaste prompts. Toepassingen krijgen eerst een link van het wetenschappelijk artikel. 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 voorzetselzinnen 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 uit rond gepersonaliseerde opmaakopties. Kurzweil, SprintPlus en AlineaSuite bieden opmaakopties aan in het instellingenscherm. Zo kunnen eindgebruikers het lettertype, -kleur, -grootte en de achtergrondkleur van de tekst aanpassen naar keuze. Als een toepassing niet over opmaakopties beschikt, stopt het experiment rond opmaakopties voor die toepassing. Tot slot test het onderzoek de mogelijkheden om het formaat van teksten aan te passen met 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. Vergelijking van taalmodellen

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

- Welk taalmodel kunnen ontwikkelaars inzetten 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?

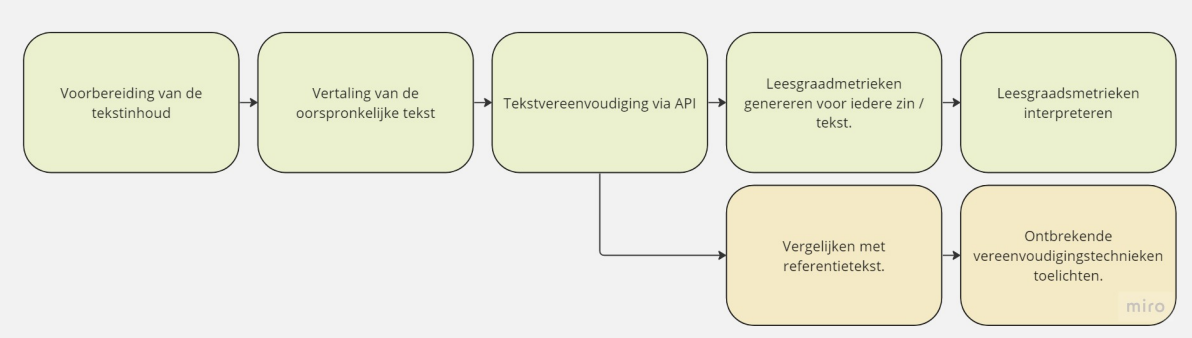
Zoals in de literatuurstudie aangegeven, beschikken ontwikkelaars over onvoldoende gespecialiseerde taalmodellen om wetenschappelijke artikelen te vereenvoudigen. Daarom vergelijkt deze onderzoeksfase alle taalmodellen uit tabel 3.6.

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

Deze onderzoeksfase bestaat uit vijf deelfasen, weergegeven op figuur 3.2. Het staat vooral stil bij de vergelijking van leesmetrieken van de oorspronkelijke wetenschappelijke artikelen, MTS-referentieteksten en ATS-teksten. Met een *mixed-methods* onderzoek kan deze onderzoeksfase taalmodellen beoordelen op machinaal en menselijk niveau. Alle scripts kan de lezer terugvinden op de GitHub-repository[1]. Het onderzoek hergebruikt dezelfde wetenschappelijke artikelen als in de requirementsanalyse, namelijk die in tabel 3.3. Om realistisch referentiemateriaal te verkrijgen, schrijven twee leerkrachten (MTSL) en twee leerlingen zonder dyslexie (MTSL2) zelf een vereenvoudiging van de twee wetenschappelijke artikelen met MTS. Deze

[1]https://github.com/Dyashen/pentimentor/tree/main/bachelorproef/scripts



Figuur (3.2)
Het gevolgde stappenplan voor de vergelijking van taalmodellen.

vier personen baseren zich op vooraf meegekregen richtlijnen, toegelicht in bijlage B.

Allereerst haalt het script de inhoud van de map op met wetenschappelijke artikelen om deze vervolgens in een tekstbestand te plaatsen, zoals weergegeven in codeblok 3.1. Hier schrijft het script tekstinhoud van het wetenschappelijk artikel over naar een nieuw tekstbestand.

```
def add_newline_after_dot(input_file, output_file):
  with open(input_file, 'r', encoding='utf-8') as file:
    text = file.read()
    text = re.sub(r'\d', '', text)
    modified_text = text.replace('.', '.\n')
    with open(output_file, 'w', encoding='utf-8') as file:
      file.write(modified_text)

    folder_path = 'scripts\pdf'
    original_scientific_papers = [f for f in os.listdir(folder_path)]

    for paper in original_scientific_papers:
      input_file =  folder_path + '/' + paper
      output_file = folder_path + '/' + 'RE_' + paper
      add_newline_after_dot(input_file, output_file)
```

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

Vervolgens vertaalt het tweede script alle zinnen naar het Engels. Eerst doorloopt het alle zinnen binnen het wetenschappelijk artikel. Daarna vertaalt het de tekstinhoud met de *deep translator* python-bibliotheek. Als resultaat ontstaat er een csv-bestand met twee kolommen: alle Nederlandstalige en alle Engelstalige zinnen van één wetenschappelijk artikel. Als separator gebruikt het csv-bestand een *pipe*-symbool en zo houdt het rekening met punten en komma's in zinnen.

```
def translate_dutch_to_english(dutch_text_file):
  with open(dutch_text_file, 'r', encoding='utf-8') as file:
    dutch_sentences = file.readlines()
    dutch_sentences = [sentence.strip() for sentence in dutch_sentences]
```

```
5
6     english_sentences = []
7     for sentence in dutch_sentences:
8       translated = GoogleTranslator(source='nl', target='en').translate(sentence)
9       english_sentences.append(translated)
10      df = pd.DataFrame({'Dutch': dutch_sentences, 'English': english_sentences})
11      df.to_csv(str(dutch_text_file).split('.')[0] + '.csv', index=False)
12
13
14    folder_path = 'scripts/pdf/'
15    original_scientific_papers = [f for f in os.listdir(folder_path)]
16
17    for paper in original_scientific_papers:
18      if paper.startswith('RE_') and paper.endswith('.txt'):
19        print(f'STARTING {paper}')
20        dutch_text_file = folder_path + paper
21        translate_dutch_to_english(dutch_text_file)
22
```

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

In een derde fase moet het script API-calls voor iedere zin of paragraaf versturen naar ieder taalmodel. Eerst slaat het script een *dictionary* op van taalmodellen om de inhoud van de wetenschappelijke artikelen te vereenvoudigen, zoals weergegeven in listing 3.3. Daarna vindt tokenisatie plaats, waarvoor het script Spacy *sentence tokenisation* en verwante *embedding models* gebruikt. Deze modellen staan in tabel 3.9. Na de tokenisatie stuurt het script API-calls naar de hosts van de taalmodellen. Het verzoek naar de *HuggingFace* (HF) API bestaat uit de parameters weergegeven in tabel 3.7. Alle HF-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.

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 *True*.

Tabel 3.7: Meegegeven parameters bij HF-requests

Zoals aangehaald door Gooding (2022) kunnen promptgebaseerde testen verschillende resultaten leveren, afhankelijk van de gegeven input. Daarom gebruikt het

script drie verschillende prompts, gebaseerd op de MTS-technieken beschreven in tabel 2.5. Tabel 3.8 visualiseert de gebruikte prompts voor de testen met het GPT-3 model.

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 voorzetselzinnen door eenvoudiger alternatieven, maar laat ze ongewijzigd als er geen eenvoudiger optie beschikbaar is.

Tabel 3.8: De GPT-3-prompts die in de vergelijkende studie aan bod komen.

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 tabel 2.11. Bij T1 zijn deze twee parameters ingebakken in de functie. Nadien verwerken de taalmodellen iedere zin uit de tekst. Tot slot geeft de HF of GPT-3 API een resultaat 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 Nederlandstalige 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
Engels	EN Core Web Medium

Tabel 3.9: Gebruikte SpaCy word-embeddings

```
def scientific_simplify(self, text, lm_key):
    try:
        API_URL = huggingfacemodels.get(lm_key)
        translated = GoogleTranslator(source='auto', target='en').translate(str(
    text))

        if lm_key == 'T1':
            result = self.query({"inputs": str('simplify: ' + str(translated)),"
    parameters": {"max_length": len(sentence)+10},"options":{"wait_for_model":True
    }}, API_URL)
        else:
            result  = self.query({"inputs": str(translated),"parameters": {"
    max_length": len(sentence)+10},"options":{"wait_for_model":True}}, API_URL)

        if 'generated_text' in result[0]:
            translated = GoogleTranslator(source='auto', target='nl').translate(
    str(result[0]['generated_text']))
            return translated
        elif 'summary_text' in result[0]:
            translated = GoogleTranslator(source='auto', target='nl').translate(
    str(result[0]['summary_text']))
            return translated
        else:
            return None
        except:
            return text

hf = HuggingFaceModels(os.getenv('huggingface-api-key'))
original_scientific_papers = [f for f in os.listdir(folder_path)]
for paper in original_scientific_papers:
    sentence_tokens = process_file(paper)
        for sentence in sentence_tokens:
            for model in huggingfacemodels.keys():
                filename = "SIMPLIFIED_"+model+'_'+paper
                with open(filename, 'a', encoding='utf-8') as f:
                    output = hf.scientific_simplify(str(sentence), model)
                    f.write(str(output))

```

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

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

- De vergelijkende studie neemt de Flesch-Reading-Ease (FRE) en Gunning FOG (FOG) scores op, want deze scores kunnen de moeilijkheidsgraad van een zin of tekst machinaal meten.

- Het aantal complexe en lange woorden kan wijzen op de gebruikte *substitution generation* van het taalmodel. De *Dale-Chall Index* bepaalt de complexiteit van een woord in deze library. Ten slotte tellen alle woorden met meer dan vier lettergrepen mee als een lang woord volgens de *readability*-library.
- Het aantal hulpwerkwoorden en vervoegingen van 'zijn' kan aanduiden op mogelijke passieve stem, wat het onderzoek van Ruelas Inzunza (2020) als 'hinderende' zinsyntax vernoemt.

Dit script resulteert in een *Pandas-dataframe* met alle leesbaarheidsmetrieken uit de *readability*-library. Uiteindelijk slaat het script de *Pandas-dataframe* op als csv-bestand. Listing 3.4 omvat de code voor fase 4. Hierin berekent het script de leesmetrieken voor iedere zin. Allereerst laadt het script twee embeddingsmodellen in, weergegeven in tabel 3.9. Daarna bouwt het lege dataframes op om de meetresultaten te kunnen opslaan. Vervolgens itereert het python-script door alle tekstbestanden om deze tekst in te lezen. Voor elke zin probeert het script leesmetrieken te verkrijgen met de *readability.getmeasures* functie. Hierbij specifieert het script welke taal de readability-library moet gebruiken. Als het script alle meetwaarden succesvol kan verkrijgen, dan maakt het script een nieuwe rij met machinaal berekende leesmetrieken. Tot slot voegt het alle rijen toe aan de DataFrame. Het script verwerpt de zinnen voor het specifieke artikel waarbij het geen leesmetrieken kan berekenen.

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

```
            'Paper': paper.split('/')[2].split('.')[0],
            'Sentence': sent.text,
            'FRE': metrics['readability grades']['FleschReadingEase'],
            'FOG': metrics['readability grades']['GunningFogIndex'],
          }

      for key, value in metrics['sentence info'].items():
        row[key] = value

      for key, value in metrics['word usage'].items():
        row[key] = value

      for key, value in metrics['sentence beginnings'].items():
        row[key] = value

      df = df.append(row, ignore_index=True)
      except Exception as e:
        print(e)

      df.to_csv('result.csv', index=False)

```

Listing 3.4: Script voor de vierde fase van de vergelijkende studie

In de laatste fase visualiseert het onderzoek de verkregen resultaten. Hierbij gebruikt het *Jupyter-notebooks* en *Matplotlib* om de verkregen resultaten voor te stellen. Allereerst gebruikt de *Jupyter-notebook* een kleinere dataframe met enkel de data van A1. Daarna groepeert het script de data op basis van modellen. Het aantal woorden per zin fungeert als geaggregeerd veld. Met Matplotlib kan het script vervolgens een eenvoudige boxplot genereren. De notebook herhaalt dit proces voor beide artikelen en voor alle leesmetrieken vermeldt in tabel 3.10. Zo illlustreert *listing* 3.5 hoe het script boxplots genereert. Deze visualiseren het woordgebruik van de verschillende taalmodellen bij A1. Alle code om grafieken te genereren kan de lezer op de GitHub-repository[2] terugvinden.

```
artikel_1 = df[(df['paper'] == 'Artikel 1 AI') & (df['FRE'] > 0)]
data = artikel_1.groupby('model')['words_per_sentence']
data_list = [group[1].tolist() for group in data]
plt.figure(figsize=(20,10))
plt.boxplot(data_list)
plt.xticks(range(1, len(data_list) + 1), data.groups.keys())
plt.title('Woorden per zin per model')
plt.xlabel('Model')
plt.ylabel('Woorden per zin')
plt.savefig('boxplot-avg-a1.png')
```

Listing 3.5: Code om een boxplot voor het aantal woorden per zin te genereren.

[2]https://github.com/Dyashen/pentimentor/blob/main/bachelorproef/scripts/PHASE_5.ipynb

Verder toont tabel 3.10 alle leesmetrieken, samen met de toegepaste visualisatietechniek. De boxplots dienen om de spreiding van de leesmetrieken te tonen. Zo kan het onderzoek achterhalen hoe gespreid de gebruikte woordenschat is op het vlak van de leesmetrieken FRE en FOG. De spreiding bij het aantal woorden per zinnen geeft het onderzoek ook weer als een boxplot. Zo kan het onderzoek de regelmaat van korte zinnen achterhalen uit een tekst. De *violinplots* dienen om de verdeling van moeilijke of lange woorden weer te geven. Tot slot maakt het onderzoek gebruik van een staafdiagram om het aantal hulpwerkwoorden of aantal zinnen met een vervoeging van het werkwoord 'zijn' te visualiseren.

Leesmetriek	**Visualisatietechniek**
FOG	Boxplot
FRE	Boxplot
Aantal woorden per zinnen	Boxplot
Aantal complexe woorden per zin volgens Dale Chall index	Violinplot
Aantal lange woorden per zin	Violinplot
Aantal gebruikte hulpwerkwoorden	Staafdiagram
Aantal zinnen met een vervoeging van het werkwoord 'zijn'	Staafdiagram

Tabel 3.10: Visualisatietechnieken voor de machinale metrieken bij de vergelijking van de vereenvoudigde teksten met de oorspronkelijke tekst en de referentieteksten.

Tenslotte komen de resultaten van de menselijke beoordeling aan bod. Deze fase van de vergelijkende studie staat stil bij aspecten die de 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	Vereenvoudigings-techniek
Acroniemen behouden	LS
Inschatting van de doelgroep	LS
Behoud van kern- en bijzaken	LS
Schrijven in tabelvorm of als opsomming	SA
Passieve zinconstructies herschrijven naar actieve zinconstructies	SS
Bronvermelding behouden	SA
Citeren en parafraseren	SS en SA.

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

3.3. Prototype voor tekstvereenvoudiging

Met de gevonden vereisten voor een taalmodel uit het moscow-schema en de geschikte taalmodellen voor gepersonaliseerde ATS kan het onderzoek een volgende stap zetten om de onderzoeksvraag te beantwoorden. De volgende sectie omschrijft de ontwikkeling van Pentimentor, namelijk een prototype voor gepersonaliseerde ATS voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Deze ontwikkeling kan ontwikkelaars helpen als rode draad bij het maken van dergelijke toepassingen. Zo beantwoordt de ontwikkeling van *Pentimentor* volgende deelvraag:

- Hoe kunnen ontwikkelaars een intuïtieve en lokale webtoepassing ontwikkelen die zowel scholieren met dyslexie als leerkrachten kan helpen bij het vereenvoudigen van wetenschappelijke artikelen met behoud van semantiek, jargon en zinsstructuren?

Voor de ontwikkeling van het prototype volgt het onderzoek 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 ontwikkelt het de backend en frontend. Daarna ontwikkelt het twee algemene componenten: een leraren -en scholierencomponent. Hierna moet het onderzoek stilstaan bij de opzet van Pentimentor.

Uiteindelijk beoordeelt het onderzoek de *Pentimentor* prototype. Het vergelijkt dit met andere toepassingen volgens het moscow-schema. Zo moet het lerarencomponent wetenschappelijke artikelen kunnen vereenvoudigen, nadat de gebruiker ATS-technieken selecteert. Daarnaast moet het scholierencomponent ondersteuning bieden aan scholieren die de verschillen tussen de originele en vereenvou-

digde tekst willen zien. Hier moeten scholieren met dyslexie in *real-time* aanpassingen kunnen maken aan de tekst en ondersteuning krijgen tijdens het begrijpend lezen van een tekst. Tot slot moeten gepersonaliseerde opmaakopties gelijk blijven over alle pagina's heen van Pentimentor.

Taalmodellen selecteren

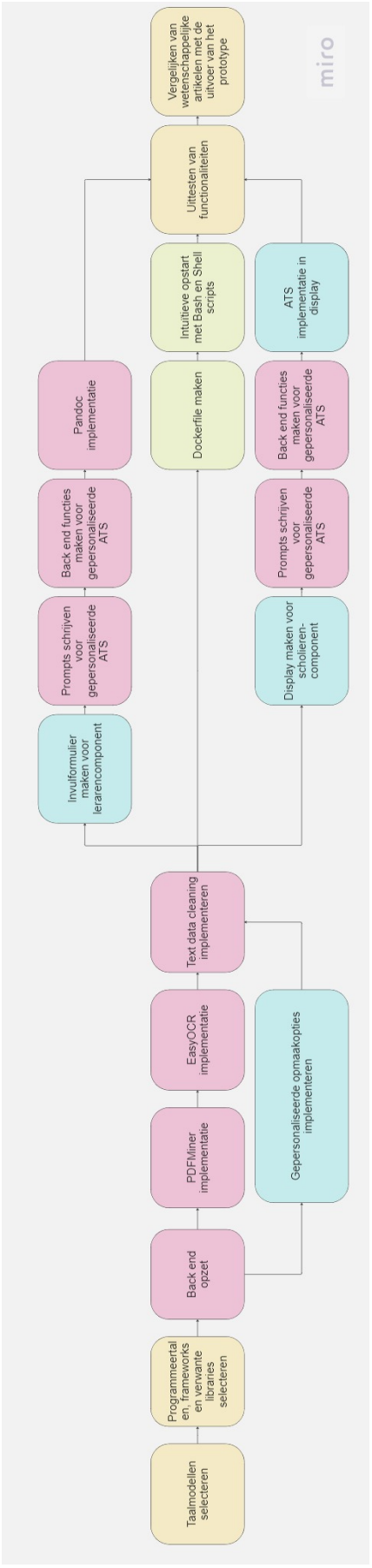
Omdat de keuze van taalmodellen verdere technologiekeuzes kan beïnvloeden, bepaalt het onderzoek eerst één of meerdere taalmodellen voor gepersonaliseerde ATS. Zo wijst de vorige onderzoeksfase uit dat GPT-3 een geschikt taalmodel is voor gepersonaliseerde ATS. Voor een abstraherende samenvatting kan *Pentimentor* één van drie taalmodellen uit de vorige onderzoeksfase gebruiken.

Programmeertalen, frameworks en verwante libraries selecteren

Voor een snelle en gratis ontwikkeling gebruikt *Pentimentor open-source* pakketten. Taalmodellen kunnen het aanspreken via *API-calls* met JS of Python. Omdat het systeem de API-sleutel moet opslaan als sessievariabele, gebeuren alle *API-calls* vanuit de backend. Verder vermeldt tabel 3.12 alle gebruikte programmeertalen. Tot slot vermeldt tabel 3.13 alle gebruikte python-libraries.

Technologie	**Functionaliteit**
Python	Dit dient voor de backend die API-calls en PoS-tags verwerkt.
JavaScript (JS)	Dit maakt de toepassing eenduidig door CLI-instructies te vervangen door knoppen.
HTML en CSS	Dit past het uiterlijk aan, naargelang de gekozen parameters van de eindgebruiker.
Jinja	Dit geeft alle Python backend data door aan de frontend.
Docker	Dit dient om Pentimentor lokaal op te kunnen starten en voorziet een lokale omgeving waarin het systeem alle software installeert.
Bash	Script om de lokale opzet op eenduidige wijze op te starten voor Linux en Mac-systemen.
Powershell	Script om de lokale opzet op eenduidige wijze op te starten voor 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 websiteframework van Pentimentor. Dit framework combineert frontend en backend.
PDFMiner	Tekstinhoud van PDF's extraheren.
LayoutParser	Selectief tekstinhoud uit PDF's extraheren.
NumPy	De *reshape*-functie vereenvoudigt de ontwikkeling om dynamisch paragrafen uit te printen.
Spacy	*PoS-tagging* en *token lemmatization*.
OpenAI	GPT-3 API aanspreken.

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

Backend opzet

Nadat het onderzoek taalmodellen, technologieën en frameworks uitkiest, start het met de frontend implementatie. Allereerst voegt het de homepagina toe. Dit dient als portaal voor de componenten. Daarna implementeert het personaliseerbare opmaakopties. Zoals aangeraden door Galliussi e.a. (2020), moeten ontwikkelaars hiermee rekening houden. Zo gebruikt Pentimentor parameters onderzocht door Rello en Baeza-Yates (2013) en Rello, Baeza-Yates, Dempere-Marco e.a. (2013). Om deze instellingen aan te bieden, voegt het onderzoek een webpagina toe. Deze HTML-pagina bevat formulieren met aanpasbare parameters. Nadat de gebruiker parameters aanpast, stuurt de pagina een POST-request naar de backend. Het verwerkt dit en slaat de ontvangen dictionary op als sessievariabele, zoals weergegeven in listing 3.6. Tot slot laadt de frontend deze opmaakopties in bij iedere wegpagina. Dit gebeurt met *window onload*, zoals weergegeven in listing 3.7. Zo hoeven eindgebruikers deze parameters niet bij iedere webpagina opnieuw aan te passen.

```
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:
```

```
    msg = str(e)
    flash(msg)
  return render_template('settings.html')
```

Listing 3.6: De backend functie die de aanpassingen uit het formulier opslaat als sessievariabele.

```
window.onload = async function () {
  var url = `http://localhost:5000/get-settings-user`;
  const response = await fetch(url, { method: 'POST' });
  var result = await response.json();
  document.body.style.fontSize        = result.fontSize+'px';
  document.body.style.fontFamily      = result.fontSettings;
  document.body.style.backgroundColor = result.favcolor;
  document.body.style.lineHeight      = result.lineHeight+'cm';
  document.body.style.wordSpacing     = result.wordSpacing+'cm';
  document.body.style.textAlign       = result.textAlign;
}
```

Listing 3.7: De onload-functie die de gepersonaliseerde opmaakopties regelt bij het inladen van een webpagina.

Naast personaliseerbare opmaakopties moet *Pentimentor* wetenschappelijke artikelen kunnen inladen en inlezen. Zo kan het met *PDF-extractors* zoals *PDFMiner* tekst uit de PDF extraheren. Niet alle *PDF-extractors* zijn foutbestendig, want zoals opgemerkt in de literatuurstudie kunnen *PDF-extractors* ook tekst verliezen tijdens dit proces. Om dit te voorkomen, biedt het prototype een OCR-optie met *LayoutParser* aan. Beide manieren vereisen functies in de backend en frontend.

Aan de frontend voegt het onderzoek checkboxes, file-input en texarea-input tags toe. Uiteindelijk stuurt het formulier een POST-request om alle meegekregen informatie door te sturen naar de backend. Na het ontvangen van de request van de frontend, handelt de Flask backend het bestand verder af zoals aangewezen in listing 3.8. Nadat de gebruiker het formulier bevestigt, geeft het de binaries van het wetenschappelijk artikel door aan de backend. Die controleert het type invoer en slaat het nadien tijdelijk *in-memory* op. Daarnaast krijgt het de keuze tussen normale of OCR-upload mee als boolean, waarbij *true* gelijk staat aan een afhandeling met LayoutParser.

```
def setup_scholars_teachers(request):
  settings = request.form
  if 'fullText' in request.form:
    text = request.form['fullText']
    langs = detect_langs(text)
    reader = Reader()
    dict_text = reader.get_full_text_site(text)
  elif 'pdf' in request.files:
    if 'advanced' not in settings:
      pdf = request.files['pdf']
      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 3.8: Koppeling tussen frontend en backend voor het inlezen van een wetenschappelijk artikel

PDFMiner implementatie

Bij een gewone PDF-extractie gebruikt de Reader-klasse PDFMiner met de functie in listing 3.9. Het itereert door alle pagina's van een opgeladen wetenschappelijk artikel. Nadien extraheert het alle tekst van een pagina en concateneert het alle opgehaalde tekst van één pagina aan een lege string. Dit proces herhaalt zich tot het geen pagina's meer kan inlezen. Tot slot resulteert dit in een string-object met alle geëxtraheerde tekst uit het wetenschappelijk artikel. Hierna moet de Reader-klasse deze tekst formatteren tot een leesbaar formaat.

```
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 3.9: Een PDF inlezen met PDFMiner

OCR implementatie

Tekst kan ontbreken na het extraheerproces met PDFMiner. Daarnaast krijgt Pentimentor als resultaat één string van alle karakters. Tot gevolg toont het deze tekst op een niet-georganiseerde manier. Daarom voorziet het prototype een tweede methode met OCR en machineleertechnieken (ML). Zo gebruikt Pentimentor het Python-pakket *Layoutparser* om bovenstaand probleem op te lossen.

In eerste instantie moet de backend alle pagina's omzetten naar een machine-interpreteerbaar formaat door ze eerst om te zetten naar afbeeldingen en vervolgens naar binaire waarden. Met behulp van LayoutParser en het Detectron2-algoritme kan Pentimentor de verschillende tekstonderdelen op elke pagina classificeren, waaronder tekst-, titel-, figuur- en tabelblokken. Omdat figuren en tabellen niet relevant zijn voor dit onderzoek, concentreert Pentimentor zich op het achterhalen van titels en tekst. Die slaat het systeem op in een array genaamd 'valid'. Daarna omkadert LayoutParser alle tekstblokken en koppelt het elk tekstblok met een unieke id-tag. Listing 3.10 illustreert het proces om de blokken op één pagina te achterhalen.

```
model = lp.Detectron2LayoutModel(
  config_path ='lp://PubLayNet/faster_rcnn_R_50_FPN_3x/config',
  label_map   ={0: "Text", 1: "Title", 2: "List", 3:"Table", 4:"Figure"},
  extra_config=["MODEL.ROI_HEADS.SCORE_THRESH_TEST", 0.8])

image = cv2.imread(img_folder + afbeelding)
image = image[...,::-1]

layout = model.detect(image)

lp.draw_box(image, layout, box_width=3)

valid = ['Text', 'Title']
text_blocks = lp.Layout([b for b in layout if b.type in valid])

h, w = image.shape[:2]

left_interval = lp.Interval(0, w/2*1.05, axis='x').put_on_canvas(image)

left_blocks = text_blocks.filter_by(left_interval, center=True)
left_blocks.sort(key=lambda b:b.coordinates[1], inplace=True)

right_blocks = [b for b in text_blocks if b not in left_blocks]
right_blocks.sort(key=lambda b:b.coordinates[1])
```

```
25
26 text_blocks = lp.Layout([b.set(id=idx) for idx, b in enumerate(left_blocks+
       right_blocks)])
27
28 lp.draw_box(image, text_blocks, box_width=3, show_element_id=True)
29
```

Listing 3.10: Een PDF inlezen met OCR

Nu beschikt Pentimentor over een array van afbeeldingen. Vervolgens moet het alle tekst uit deze blokken kunnen extraheren. Hiervoor raadt de documentatie van Layoutparser een OCR-methode aan. Listing 3.11 illustreert dit. Zo gebruikt Pentimentor de Python-bibliotheek *Tesseract* om de titels en tekst te extraheren uit de afbeeldingen. Omdat het systeem niet bijhoudt wat titel of tekstblok is, voegt het systeem een string toe aan iedere tekstblok. Zo resulteert deze functie in een array met alle teruggevonden titel- en tekstblokken.

```
1 language = language
2 ocr_agent = lp.TesseractAgent(languages=language)
3
4 full_text = []
5
6 for block in text_blocks:
7   segment_image = (block.pad(left=5, right=5, top=5, bottom=5).crop_image(image))
8   text          = ocr_agent.detect(segment_image)
9   block.set(text=text, inplace=True)
10
11 for t in text_blocks:
12   if(t.type == 'Title'):
13     full_text.append('title:' + str(t.text))
14   else:
15     full_text.append(t.text)
16   return full_text
```

Listing 3.11: Tekst extraheren uit de geparsete inhoud.

Geëxtraheerde inhoud formatteren naar een webpagina.

Na het extraheren van tekst volgt het formatteren ervan. Hier zet het systeem de tekst om naar het gewenste formaat voor de *frontend*. In eerste instantie transformeert de *backend* de geëxtraheerde tekst naar *arrays* van zinnen met behulp van Spacy *word embeddings* en *sentence embeddings*. Listing 3.12 illustreert de werking van deze techniek. Zo bundelt de backend een parameteriseerbaar aanal zinnen met behulp van *Numpy reshape*. Om de Part-of-Speech (PoS)-tag bij het respectievelijke woorde te behouden, gebruikt de backend een sleutelpaarstructuur. Elke sleutel verwijst naar een woord in een zin, terwijl de waarde naar de bijbehorende PoS-tag verwijst. Het onderzoek focust zich op de ontwikkeling voor de vereenvoudiging van Nederlandstalige of Engelstalige wetenschappelijke artikelen. Daarom laadt Pentimentor hoogstens twee embeddingsmodellen in. Tabel

3.9 vermeldt deze embeddingsmodellen. Standaard maakt Pentimentor gebruik van een Engelstalig embeddingsmodel als de backend de taal niet kan herkennen of als de taal niet voorkomt in een *dictionary*.

```
def get_full_text_site(self, full_text):
  try:
    lang = detect(full_text)
  except:
    lang = 'en'

  if lang in dict:
    nlp = spacy.load(dict.get(lang))
  else:
    nlp = spacy.load(dict.get('en'))

  full_text = str(full_text).replace('\n', ' ')

  doc = nlp(full_text)
  sentences = []
  for sentence in doc.sents:
    sentences.append(sentence)

  pad_size = SENTENCES_PER_PARAGRAPH - (len(sentences) % SENTENCES_PER_PARAGRAPH
  )
  padded_a = np.pad(sentences, (0, pad_size), mode='empty')
  paragraphs = padded_a.reshape(-1, SENTENCES_PER_PARAGRAPH)

  text_w_pos = []
  for paragraph in paragraphs:
    paragraph_w_pos = []
    try:
      for sentence in paragraph:
      dict_sentence = {}
      for token in sentence:
        dict_sentence[token.text] = str(token.pos_).lower()
        paragraph_w_pos.append(dict_sentence)
        text_w_pos.append(paragraph_w_pos)
    except:
      pass

  return text_w_pos
```

Listing 3.12: Het formatteren van de tekst naar een formaat voor de website.

Beide componenten moeten het artikel in een aangepaste pagina tonen. Daarom maakt deze fase gebruik van Jinja2, de *templating engine* van Flask. Hiermee kan de frontend de array uit de vorige fase itereren en vervolgens uitprinten. Ieder woord koppelt de backend met een PoS-tag. Daarom koppelt de frontend iedere zin met de overkoepelende span-tag *sentence*. Daarna koppelt het ieder woord met de klasse *nouns*, *adjectives* of *verbs*. Andere woorden, zoals conjuncties, kop-

pelt het systeem met de klasse *other*.

```
{% for paragraph in pdf %}
  <p class="left-side">
    {% for sentence in paragraph %}
    <span class="sentence">
      {% for word in sentence %}
      {% if sentence[word] != 'space' %}
      <span class={{sentence[word]}}>{{word}}</span>
      {% endif %}
      {% endfor %}
    </span>
  {% endfor %}
  </p>
{% endfor %}
```

Listing 3.13: 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 tekstinhoud kunnen manipuleren, om vervolgens opties voor gepersonaliseerde ATS te selecteren. Figuur 4.20 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 webinhoud sleutelparen 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	Frontend aanpassing met *eventlistener*. De tekstkleur van het aangeduide type woorden verandert naar het gekozen kleur.
Zinnen verwijderen	*Frontend* filter aangesproken door een *eventlistener*.
Woord toevoegen aan de woordenlijst	Een *eventlistener* handelt de functionaliteit af. Het prototype slaat woorden en hun context tijdelijk op. Het formulier houdt dit bij en geeft het vervolgens mee bij het indienen. Deze woorden en hun respectievelijke zin van voorkomen dienen om de woordenlijst op te vullen in het gegenereerde PDF of Word-bestand.

Tabel 3.14: Alle beschikbare functionaliteiten in het lerarencomponent.

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	Schrijf deze zin eenvoudiger // 'zin'
Gepersonaliseerde ATS	Schrijf deze zin eenvoudiger volgens de colgende criteria // 'zin' // criteria: 'criteria'
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 definities van woorden te genereren met GPT-3.

Functies voor gepersonaliseerde ATS in het lerarencomponent

Gebruikers kunnen Pentimentor op twee manieren gebruiken. Zij kunnen het volledige artikel automatisch laten vereenvoudigen of samenvatten, of specifiek fragmenten markeren om een gepersonaliseerde vereenvoudiging of samenvatting te laten maken. Zo moet Pentimentor een nieuwe doorlopende tekst kunnen genereren, als opsomming of tabel laten herschrijven of woordenlijsten genereren. Hiervoor moet het twee zaken bijhouden: de gemarkeerde tekst en de optie die de

gebruiker wenst te kiezen. Hiervoor gebruikt het onderzoek een sleutelpaarstructuur. Zo biedt *localstorage* een tijdelijk persistente methode. De frontend moet de *localstorage* opvullen met gemarkeerde tekst en na gebruik opnieuw leegmaken. Als het systeem dit niet leegmaakt, dan neemt het systeem overblijfselen van vorige artikelen over. Om eindgebruikers van hun keuzes bewust te maken, markeert de frontend deze tekst met een kleur. Verder maakt de frontend de eindgebruiker bewust welke kleur bij welke transformatie hoort. Tot slot toont listing 3.14 de werking van deze methode.

```
function addTextWithParagraph() {
  text = window.getSelection().toString();
  if (text == "") {
    return;
  } else {
    var fieldset = document.querySelector(".personalized");
    var inputs = fieldset.querySelectorAll("input");
    var values = [];
    for (var i = 0; i < inputs.length; i++) {
      if (inputs[i].type === "checkbox" && inputs[i].checked) {
        values.push(inputs[i].value);
      }
    }

    var selection = window.getSelection();
    if (selection.rangeCount > 0) {
      var range = selection.getRangeAt(0);
      var commonAncestor = range.commonAncestorContainer;
      var selectedElements = commonAncestor.getElementsByClassName("sentence");
      var selectedTags = [];
      for (var i = 0; i < selectedElements.length; i++) {
        var elementRange = document.createRange();
        elementRange.selectNodeContents(selectedElements[i]);

        if (range.intersectsNode(selectedElements[i])) {
          selectedTags.push(selectedElements[i]);
        }
      }
    }

    fullText = "";

    for (var i = 0; i < selectedTags.length; i++) {
      var tag = selectedTags[i];
      if (values.includes('summation')) {
        tag.style.backgroundColor = "#F0FFF0";
      } else if (values.includes('glossary')){
        tag.style.backgroundColor = "#F5DEB3";
      } else if (values.includes('table')) {
        tag.style.backgroundColor = "#FAFAD2";
      } else {
```

```
42          tag.style.backgroundColor = "#F5F5DC";
43        }
44
45        var tagText = tag.textContent;
46        tagText = tagText.split('\n').join('').replace(/:/g, "");
47        fullText += tagText;
48      }
49      localStorage.setItem(fullText, values);
50    }
51  }
52
53  function getAllLocalStorageValues() {
54    var localStorageValues = {};
55    for (var i = 0; i < localStorage.length; i++) {
56      var key = localStorage.key(i);
57      var value = localStorage.getItem(key);
58      localStorageValues[key] = value;
59    }
60    return localStorageValues;
61  }
```

Listing 3.14: Implementatie rond localstorage en tekst markeren.

Verbinding met GPT-3

Om de gemarkeerde tekst te kunnen verwerken, moet het systeem de inhoud van de *localstorage* doorsturen naar de backend. Vervolgens moet het ieder sleutel-paar overlopen. Iedere sleutel stelt een fragment voor dat de backend moet vereenvoudigen of samenvatten. Als de tekst het maximum aantal tokens overschrijdt, splitst de backend deze tekst op in een aantal delen dat gelijk staat aan de gelijke deling van het maximum aantal tokens, zoals weergegeven in listing 3.15.

```
1  for key, value in full_text.items():
2    new_sentence = []
3    for word in key.split(" "):
4      if word != '':
5        new_sentence.append(word)
6    sent             = " ".join(new_sentence)
7    aantalDelingen   = (len(sent)  // 1000) + 1
8    perHoeveel       = (len(sent)) // aantalDelingen
9    sent = [sent[i:i+perHoeveel] for i in range(0, len(sent), perHoeveel)]
10
11   for s in sent:
12     new_sent, prompt = gpt.personalised_simplify(s, str(value).split(','))
13     if 'summation' in str(value).split(','):
14       doc_creator.generate_summary_w_summation(new_sent)
15     elif 'table' in str(value).split(',') or 'glossary' in str(value).split(','):
16       doc_creator.generate_simplification(new_sent)
17     else:
18       doc_creator.generate_simplification(new_sent)
```

Listing 3.15: Alle gemarkeerde tekstblokken uit de localstorage aflopen en doorsturen naar het markdown document.

Vervolgens moet de *backend* deze prompts naar de GPT-3 API sturen. Zo itereert het over ieder sleutelpaar en bouwt het daarmee prompts op. Deze bestaat uit gekozen opties en gemarkeerde tekst van de gebruiker. Naast de prompt dient de backend ook parameters mee te geven. Deze omvatten alle waarden opgelijst in tabel 3.16. Omdat formaatwijzigingen specifieke prompts vereisen, bestaat de prompt uit extra klemtonen om het resultaat in markdowncode terug te krijgen. Als de *backend* geen formaatwijziging moet uitvoeren, dan stuurt het generieke prompts met daarin een opsomming van alle kenmerken die de eindgebruiker wilt zien in de vereenvoudigde tekst. Listing 3.16 toont de code die de backend voor één gemarkeerd tekstblok uitvoert. Deze instructies herhaalt het voor ieder tekstblok dat de gebruiker heeft gemarkeerd.

```
def personalised_simplify(self, sentence, personalisation):
  if 'table' in personalisation:
  prompt = f"""
    Herschrijf de tekstinhoud maar in een tabel, gebruik twee kolommen naar keuze;
     schrijf dit in markdowncode.
    ///
    {sentence}
    """
  elif 'glossary' in personalisation:
    prompt = f"""
    Maak een woordenlijst (max 5 woorden) in tabelvorm van het gebruikte jargon
    uit deze tekst; schrijf dit in markdowncode. ///
    {sentence}
    """
  else:
    prompt = f"""
    Vereenvoudig de zinnen met de volgende kenmerken: {", ".join(personalisation)}
    ///
    {sentence}
    """

  try:
    result = openai.Completion.create(prompt=prompt,temperature=0,max_tokens=len(
    prompt),model=COMPLETIONS_MODEL,top_p=0.9,stream=False)["choices"][0]["text"].
    strip(" \n")
    if 'summation' in personalisation:
      result = result.split('.')
    elif 'table' in personalisation or 'glossary' in personalisation:
      result = result
    else:
      result = result
    return result, prompt
  except Exception as e:
```

```
30      return str(e), prompt
```

Listing 3.16: API-calls versturen naar de GPT-3 API.

Pandoc implementatie

Ten slotte moet het prototype de *plain-text* van vereenvoudigde tekstinhoud en de woordenlijsten in een pdf of docx-bestand gieten. Zo genereert de Creator-klasse pdf en docx-documenten volgens de meegegeven opmaakopties. Het genereren van pdf en docx-documenten gebeurt met 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 docx-document converteert. Daarvoor is een YAML-header nodig die de elementen, beschreven in tabel 3.17, moet bevatten.

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.

Allereerst bouwt het prototype een markdown-bestand met daarin de YAML-header. Zo bouwt listing 3.17 de YAML-header op. 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"title: {title}\n")
9       f.write(f"mainfont: {chosen_font}.ttf\n")
10      f.write(f"titlefont: {chosen_title_font}.ttf\n")
11      f.write(f'date: {DATE_NOW}\n')
12      f.write(f'document: article\n')
13      f.write(f'geometry: margin={margin}cm\n')
```

```
14        f.write(f'fontsize: {fontsize}pt\n')
15        f.write('header-includes:\n')
16        f.write(f'- \spaceskip={word_spacing}cm\n')
17        f.write(f'- \\usepackage{{setspace}}\n')
18        f.write(f'- \{type_spacing}\n')
19        f.write("---\n")
```

Listing 3.17: 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 prototype vervolgens 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. Listing 3.18 toont deze functie. 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 3.18: Een woordenlijst genereren met de Writer-klasse.

Daarna vult het script het markdownbestand op met de vereenvoudigde tekstinhoud, zoals weergegeven in listing 3.19. Deze tekst is gesplitst door de titels die de leerkracht heeft gekozen. Daarna slaat het script de vereenvoudigde tekst op in een *dictionary*-structuur. Vervolgens print het script de vereenvoudigde tekst uit naar het markdownbestand 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 3.19: Een doorlopende tekst toevoegen aan het markdownbestand met de Writer-klasse.

Het prototype voert een andere functie uit als de tekst een opsomming moet zijn in het uitvoerbestand. Listing 3.20 toont deze functie. Als de leerkracht een opsom-

ming wenst, dan dient de titel nog steeds al separator. Enkel zal het script de zinnen uitprinten als opsomming conform aan de markdownsyntax. 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.

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

Listing 3.20: Een opsomming toevoegen aan het markdownbestand met de Writer-klasse.

Tenslotte maakt het script de pdf en docx-bestanden van de vereenvoudigde teksten. Beide bestanden maken gebruik van dezelfde YAML-header. Listing 3.21 gebruikt alle bovenstaande functies indien nodig. Daarna genereert Pandoc de pdf en docx-bestanden en 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.

```
def create_pdf(self, title, margin, list, full_text, fonts, word_spacing,
    type_spacing, summation):
  if title is not None:
    self.create_header(title=title, margin=margin, fontsize=14, chosen_font=fonts
    [0], chosen_title_font=fonts[1], word_spacing=word_spacing, type_spacing=
    type_spacing)
  else:
    self.create_header(title='Vereenvoudigde tekst', margin=0.5, fontsize=14,
    chosen_font=fonts[0], chosen_title_font=fonts[1], word_spacing=word_spacing,
    type_spacing=type_spacing)

  if len(list) != 0:
    self.generate_glossary(list=list)

  if summation:
    self.generate_summary_w_summation(full_text=full_text)
  else:
    self.generate_summary(full_text=full_text)

  pypandoc.convert_file(source_file=markdown_file, to='docx', outputfile=docx_file
    ,    extra_args=["-M2GB", "+RTS", "-K64m", "-RTS"])
```

Functionaliteit	JS/GPT-3
Zinnen verwijderen	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.

```
16  pypandoc.convert_file(source_file=markdown_file, to='pdf', outputfile=pdf_file,
        extra_args=['--pdf-engine=xelatex'])
17  with zipfile.ZipFile(zip_filename, 'w') as myzip:
18    myzip.write(pdf_file)
19    myzip.write(docx_file)
```

Listing 3.21: 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 hierop kunnen scholieren zelf een prompt schrijven. Het systeem gebruikt deze prompt met de gemarkeerde tekst als input voor GPT-3.

Backend functies schrijven voor gepersonaliseerde ATS

Om zinnen in de doorlopende tekst te verwijderen, moet de frontend over de nodige span-tags beschikken. Bij het inlezen van de PDF voert de backend dit al uit, zoals verwezen in listing 3.12. Daarom hoeft het onderzoek geen nieuwe functie in de backend te schrijven. Vervolgens moet de backend over een functie beschikken

om een zin te vereenvoudigen. Het lerarencomponent gebruikt al dergelijk functie. Daarmee hergebruikt het de functie in listing 3.16.

Zinnen baseren op een zelfgemaakte prompt moet de *backend* echter wel toevoegen. Zo gebruikt de backend de functie in listing 3.22 om een *custom prompt* te verwerken. Deze functie stuurt de oorspronkelijke prompt, samen met de context, door naar de GPT-3 API.

```
def personalised_simplify_with_prompt(self, sentences, personalisation):
  try:
    result = openai.Completion.create(
      prompt=personalisation,
      temperature=0,
      max_tokens=len(personalisation)+len(sentences),
      model=COMPLETIONS_MODEL,
      top_p=0.9,
      stream=False
    )["choices"][0]["text"].strip(" \n")
    return result, personalisation
  except Exception as e:
    return str(e), personalisation
```

Listing 3.22: Een API-call sturen naar GPT-3 met een custom prompt.

Tot slot heeft de backend als een functie die een woord opzoekt met GPT-3. Deze functie staat beschreven in listing 3.23 en de backend kan deze functie zonder problemen hergebruiken. Tot slot hoeft de *backend* geen nieuwe functie te krijgen om woordmarkering af te handelen.

```
def look_up_word_gpt(self, word, context, lemma, pos):
  try:
    prompt = f"""
    Geef een eenvoudige definitie:
    woord: {word} /// lemma: {lemma} /// pos: {pos} /// context: {context}
    ///
    """

    result = openai.Completion.create(
        model=COMPLETIONS_MODEL,
        prompt=prompt,
        temperature=0,
        max_tokens=80,
        top_p=0.9,
        stream=False
        )["choices"][0]["text"].strip(" \n")

    return result, word, prompt
  except Exception as e:
    return 'error', str(e), str(e)
```

Listing 3.23: Een API-call sturen naar GPT-3 om de definitie van een woord op te halen.

Frontend implementatie voor ATS functionaliteiten

Backend logica kan voorlopig enkel via commandline worden geraadpleegd. Zo moeten deze functies ook op een eenduidige manier beschikbaar zijn voor eindgebruikers. Daarom maakt Pentimentor gebruik van Javascript om de *backend* logica met eenduidige handelingen te kunnen aanspreken. Dit gebeurt op verschillende manieren.
Allereerst kan de *frontend* zonder hulp van de *backend* zinnen verwijderen. Zo bundelt het alle woorden in een zin met een span-tag van de 'sentence' klasse. Als de gebruiker dergelijk span-tags aanklikt, dan verwijdert het de gekozen span-tag.

Eerst slaat JS de gemarkeerde tekst en meegekregen ATS-opties op en geeft deze door aan de backend met een API-call. Vervolgens verwerkt de backend 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 specifieert 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 frontend alle zinnen doorlopen en uitprinten tussen twee li-tags.

Listing 3.24 toont de aanpak om via de frontend een woord, pos-tag en definitie toe te voegen aan de woordenlijst tabel. De frontend handelt enkel aanvragen voor werkwoorden, adjectieven, hulpwerkwoorden en zelfsandige naamwoorden af. Eenmaal de scholier op een woord drukt, stuurt de frontend een aanvraag naar de backend. Hiervoor stuurt de frontend de context en het woord naar de backend. De frontend 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",
```

```
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ word: word, sentence: context }),
      });
      result = await response.json();

      if (result.result == "error") {
        alert("Incorrect API key provided: " + result.word);
      } else {
        var pos_tag = result.result.split('|')[0];
        var definition = result.result.split('|')[1];
        let table = document.querySelector(".table-glossary");
        let newRow = table.insertRow(-1);
        let cell1 = newRow.insertCell(0);
        let cell2 = newRow.insertCell(1);
        let cell3 = newRow.insertCell(2);
        cell1.innerHTML = result.word;
        cell2.innerHTML = pos_tag;
        cell3.innerHTML = definition;
      }
    });
  });
});
```

Listing 3.24: Een woord aan de woordenlijst toevoegen in het scholierencomponent.

Tot slot moet de frontend specifieke woorden kunnen markeren. De frontend beschikt al over de PoS-tags. Zo kan de frontend, met de JS-functie in listing 3.25, deze woorden een specifieke kleur geven als markering. De listing toont enkel het markeren van zelfstandige naamwoorden. Daarnaast kan de frontend ook adjectieven uit de tekst verwijderen zonder taalmodel. Zo hoeft de JS-functie enkel de span-tags van de klasse 'adj' verwijderen uit de document.

```
const nouns = document.getElementById('noun-show');

nouns.addEventListener('change', function () {
  if (this.checked) {
    const color = document.getElementById('colorForNouns').value;
    const elements = document.querySelectorAll("span.noun");
    elements.forEach(function (element) {
      element.style.color = color;
    });
  } else {
    const elements = document.querySelectorAll("span.noun");
    elements.forEach(function (element) {
      element.style.color = "black";
    });
  }
});
```

Listing 3.25: Zelfstandige naamwoorden in het scholierencomponent markeren.

3.3.2. De opzet voor een lokale webtoepassing.

Eindgebruikers kunnen voorlopig enkel Pentimentor via *commandline* opstarten. Het online plaatsen valt buiten de *scope* van dit onderzoek, maar Docker en zelfgeschreven scripts bieden een eenduidige opstart. Hierbij moet de eindgebruiker geen technische vaardigheden gebruiken. Omdat Pentimentor enkel API's aanspreekt, werkt het met één Docker-container. Listing 3.26 toont de zelfgeschreven code van de Dockerfile. Om de installatie van Python soepel te laten verlopen, bouwt het onderzoek een lijst van nodige python-bibliotheken op. Daarnaast moet het systeem de nodige Spacy word-embeddings en *LayoutParser* pakketten installeren. Dat gebeurt met aparte commando's in de Dockerfile. Met dit proces starten kan de versies tegen het einde van de ontwikkeling gedateerd maken. Ten laatste moeten ook alle lettertypen zich in de lokale map bevinden, want Pandoc moet over alle lettertypen beschikken om een docx-bestand te maken.

```
1  FROM ubuntu
2
3  WORKDIR /app
4
5  RUN apt-get update
6
7  RUN apt-get install -y git python3-pip \
8  && apt install -y cmake tesseract-ocr libtesseract-dev poppler-utils
9
10 RUN pip install --upgrade pip
11
12 RUN pip install layoutparser \
13 && pip install "layoutparser[effdet]" layoutparser torchvision && pip install "git
       +https://github.com/facebookresearch/detectron2.git@v0.5#egg=detectron2" \
14 && pip install "layoutparser[paddledetection]" \
15 && pip install "layoutparser[ocr]" \
16 && pip install pytesseract easyocr pdf2image flask spacy
17
18 RUN export LD_LIBRARY_PATH=/usr/lib/wsl/lib:$LD_LIBRARY_PATH
19
20 RUN python3 -m spacy download nl_core_news_md
21
22 RUN apt install -y cmake tesseract-ocr libtesseract-dev poppler-utils
23
24 RUN pip uninstall -y Pillow && pip install Pillow==9.5.0
25
26 COPY . .
27
28 CMD ["python3", "-m", "flask", "run", "--host=0.0.0.0", "--port=5000"]
```

Listing 3.26: Dockerfile voor Pentimentor.

Tot slot kan niet iedereen eenvoudig een Docker-container opstarten. Daarom voegt het onderzoek een optioneel scriptbestand toe. Een scriptbestand in Powershell, zoals weergegeven in listing 3.27, of Bash zoals weergegeven in listing 3.28,

maakt de opstart van deze webapplicatie intuïtiever dan via *commandline*. Hiermee kunnen MacOS, Linux en Windows-gebruikers Pentimentor installeren. De automatische installatie van Docker valt buiten het doel van het script, maar het systeem prompt gebruikers om eerst Docker te installeren.

```
@echo off
cd web-app
docker stop text-application-prototype
docker rm text-application-prototype
docker rmi text-app
docker build -t text-app .
docker run --name text-application-prototype --network webapp_simplification -d -p
    5000:5000 text-app
```

Listing 3.27: Script voor het opstarten van de Docker-container voor Windows-gebruikers

```
#!/bin/sh
cd web-app || exit
docker stop text-application-prototype
docker rm text-application-prototype
docker rmi text-app
docker build -t text-app .
docker run --name text-application-prototype --network webapp_simplification -d -p
    5000:5000 text-app
```

Listing 3.28: Script voor het opstarten van de Docker-container voor Unix-gebruikers

3.3.3. Experimenten met Pentimentor en vergelijkingen met bestaande toepassingen.

Na de ontwikkeling van Pentimentor voert het onderzoek twee testen uit. Zo kan het achterhalen of Pentimentor voldoet aan de opgestelde functionaliteiten uit de requirementsanalyse, weergegeven in tabel 3.4. Alle experimenten gebruiken de wetenschappelijke artikelen uit tabel 3.3. Daarnaast gebruiken ze de parameters beschreven in tabel 3.19. Tot slot vergelijkt het onderzoek de uitvoer van Pentimentor met de oorspronkelijke wetenschappelijke artikelen en referentieteksten door MTS.

Parameter	Gekozen variabele
Standaardlettertype	Arial
Lettertype voor titel	Arial
Regeleinde	Anderhalve
Woordspatiëring (in cm)	0.5
Documentmarge (in cm)	3
Schrijven als	Opsomming

Tabel 3.19: Gekozen parameter voor experimenten.

4

Resultaten

In dit hoofdstuk overloopt het onderzoek de resultaten van alle onderzoeksmethoden. Deze omvatten de requirementsanalyse, de vergelijking van taalmodellen en de ontwikkeling van *Pentimentor*. Allereerst bespreekt dit hoofdstuk de resultaten van de requirementsanalyse door het moscow-schema op de uitgeteste tools toe te passen. Daarna staat het onderzoek stil bij de verkregen machinale en menselijke resultaten bij de vergelijking van vier taalmodellen. Met deze resultaten kan het onderzoek een geschikt taalmodel voor personaliseerbare *automatic text simplification* (ATS) uitkiezen. Tot slot bespreekt het onderzoek het Pentimentor en vergelijkt het dit met vergelijkbare tools op basis van functionaliteiten en verschillende uitvoer.

Verder bevatten de resultaten enkel de belangrijkste grafieken. Alle resultaten slaat het onderzoek op in csv-bestanden. Deze bestanden kan de lezer terugvinden op de bijhorende GitHub-repository[1].

4.1. Ontbrekende functies bij AI-toepassingen

Tabel 4.1 toont de resultaten van de requirementsanalyse.

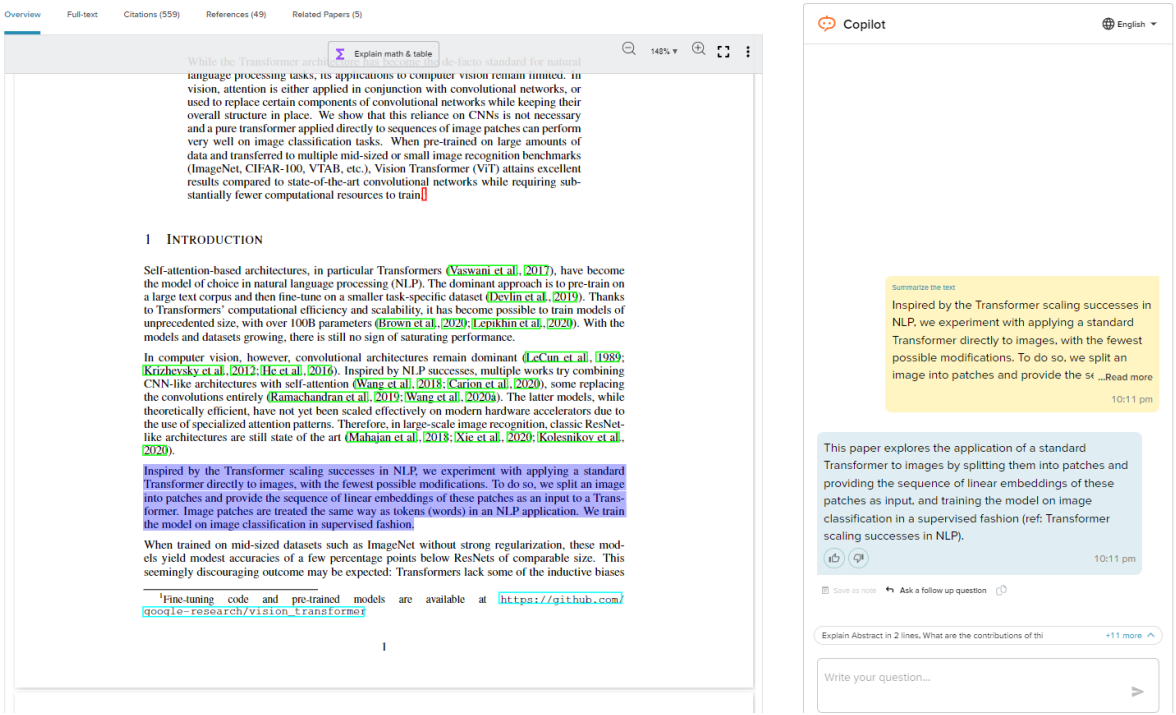
[1]https://github.com/Dyashen/pentimentor/blob/main/bachelorproef/STATISTIEKEN_BACHELORPROEF_ATS.csv

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	-	-	-	X	-	X	P3	P3
Voorzetseluitdrukkingen aanpassen	-	-	-	-	-	-	P3	P3
Samengestelde werkwoorden aanpassen	-	-	-	X	-	X	P3	P3
Actieve stem toepassen	-	-	-	-	-	-	-	-
Enkel regelmatige werkwoorden gebruiken	-	-	-	-	-	-	P3	P3
Achtergrondkleur aanpassen	X	X	X	-	-	-	-	-
Woord- en karakterspatiëring aanpassen	-	X	X	-	-	-	-	-
Consistente lay-out	X	X	X	-	-	-	P1-6	P1-6
Duidelijk zichtbare koppenstructuur	X	X	X	-	-	-	X	X
Huidige positie benadrukken	X	X	X	-	-	-	-	-
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	-	-
Artikel opladen als *plain-text*	-	-	-	X	-	X	P1-6	P1-6
Artikel opladen via link	-	-	-	-	X*	-	P1-6*	-

Tabel 4.1: Afgetoetste criteria volgens de experimenten.

Experimenten op de must-have functionaliteiten

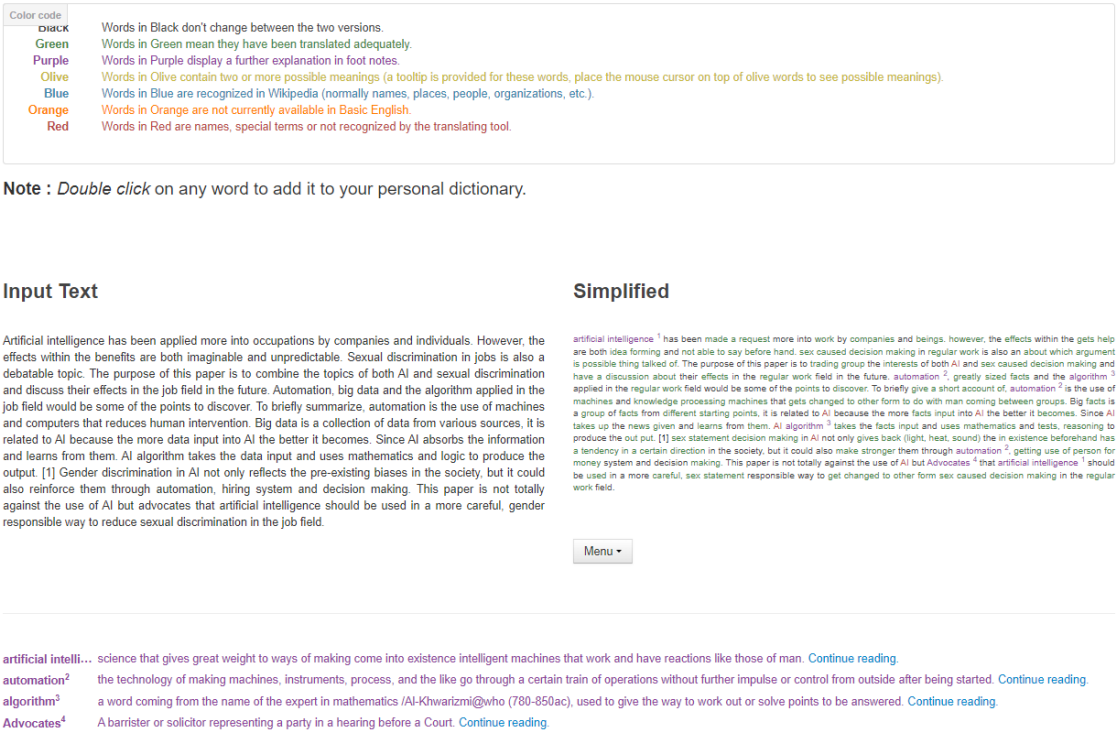
Allereerst wijzen de experimenten uit dat niet alle toepassingen de vermelde personaliseerbare opmaakopties aanreiken. Zo beschikken enkel E1, E2 en E3 over deze opties. Andere tools bieden enkel een statische webweergave aan. Verder bieden alle uitgeteste tools een methode aan om eenduidig pdf-bestanden op te slaan, met uitzondering op O1, O4 en O5. Deze uitvoerbestanden ontbreken echter een duidelijke titelstructuur en deze kunnen eindgebruikers niet eenduidig als word-bestand raadplegen. Verder kunnen gebruikers met de geteste toepassingen een wetenschappelijk artikel als pdf opladen. Uitzonderlijk laten O4 en O5 dit niet toe. Zij werken enkel met tekstinvoer. Figuur 4.1 toont de werking van toepassing O2. Deze figuur toont hoe een eindgebruiker de tekst van het oorspronkelijk pdf-bestand kan selecteren. Vervolgens kunnen zij dit eenvoudig laten samenvatten door de toepassing. Tot slot kunnen zij uitzonderlijk met O2 en O5 online wetenschappelijke artikelen via URL opladen.



Figuur (4.1)
Informatie opvragen van een wetenschappelijk artikel met SciSpace

Verder moeten toepassingen een moeilijk woord in een doorlopende tekst kunnen aanpassen. Specifiek kunnen O1, O3, O4 en O5 een annotatie toevoegen aan moeilijke woorden, maar dit gebeurt alleen als er daarvoor geen geschikt synoniem beschikbaar is. Zo illustreert figuur 4.2 hoe O1 een extra definitie als annotatie kan geven. Hoewel dit het taalniveau niet kan aanpassen, toch laat figuur 4.3 zien hoe O3 dit probeert in te schatten met een *rewordifying level*. Bovendien kunnen O4 en O5 de doelgroep aanpassen afhankelijk van de *prompts*. Andere uitgeteste tools to-

nen niet hoe zij de doelgroepinschatting maken. Geen tool laat gebruikers toe om een vooraf opgestelde woordenlijst met moeilijke woorden mee te geven waarop de tool zich kan baseren.



Figuur (4.2)
Schermafbeelding van de tekstanalyse bij Simplish na een tekstvereenvoudiging.

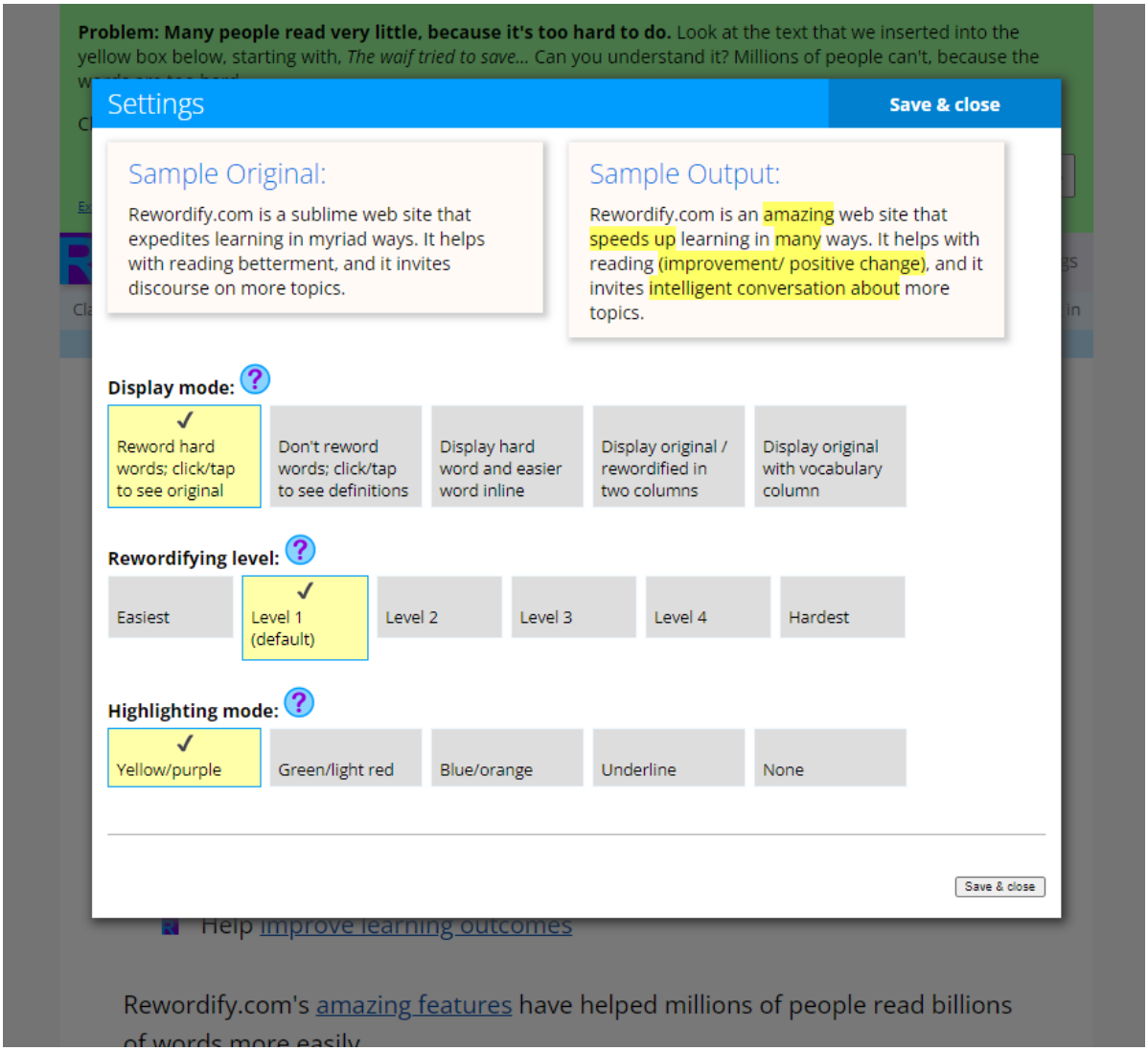
Daarnaast kunnen toepassingen ook handmatig een woordenlijst maken. Zo kunnen E1, E2, E3 en O1 een woorden- of synoniemenlijst opstellen. Bij deze toepassingen selecteren gebruikers handmatig moeilijke woorden. Die moet de toepassing vereenvoudigen of een definitie ervan ophalen. Uitzonderlijk O1 geeft de woordsoort mee. Verder kunnen gebruikers niet bepalen uit welke bron de definitie moet komen, bijvoorbeeld een online woordenboek. Enkel E1 laat dit toe.

Verder kunnen E1, E2 en E3 geen syntactische vereenvoudiging toepassen op de oorspronkelijke tekst. Overige uitgeteste tools kunnen zinnen inkorten door deze te splitsen. Geen van de uitgeteste tools kunnen de geselecteerde tekst automatisch naar een actieve vorm schrijven. In tegenstelling tot O4 en O5 die dit wel kunnen doen, maar enkel als de tool een onderwerp in de prompt meekrijgt.

Tot slot slagen O2, O4 en O5 erin om de tekst te herschrijven als opsomming. Zo doet O2 dit automatisch, vergeleken met O4 en O5 die deze vraag expliciet in hun prompt moeten krijgen. De andere uitgeteste tools kunnen dit niet automatisch doen.

Should-haves

Allereerst tonen O1 en O3 automatisch leesgraadmetrieken nadat die een vereenvoudiging of samenvatting maken van de oorspronkelijke tekst. Zo tonen figuren 4.2 en 4.3 een voorbeeld van deze analyse. Deze analyse toont het aantal zinnen en complexe/lange woorden voor zowel het oorspronkelijk als het vereenvoudigde artikel. Andere uitgeteste tools tonen dit niet.



Figuur (4.3)
Tekstanalyse met *Rewordify*.

Verder kan het onderzoek niet afleiden of de uitgeteste tools een OCR-techniek gebruiken. Wel gebruikt O2 een andere inleestechniek dan de andere tools. Zo kan het de twee gebruikte wetenschappelijke artikelen inlezen met een identieke layout als het oorspronkelijk artikel. Daarna markeert de gebruiker enkel aanpassingen in het artikel, terwijl alle uitvoer van het taalmodel rechts in beeld komt. Echter, de gebruiker kan deze aanpassing niet afleiden uit de oorspronkelijke tekst, in tegenstelling tot O1 en O3. Die tonen wel de verschillen tussen het oorspronkelijk en

het vereenvoudigd artikel aan de eindgebruiker.

Could-haves

Verschillende geteste tools maken gebruik van gebruikerfeedbacktechnieken in de vorm van *pop-ups*, zoals bij een aanpassing van de tekst. O4 en O5 vormen hierop een uitzondering, omdat zij deze technieken niet bevatten. Uitzonderlijk O4 en O5 kunnen tekst omzetten naar een tabelformaat, maar dit gebeurt alleen na een expliciete prompt. Ze hebben de mogelijkheid om tekst te interpreteren en deze in een tabel te gieten. Zo maken ze gebruik van een 2 op 2 tabel, maar de gebruiker kan het aantal kolommen en rijen instellen via de prompt.

Geen van de geteste tools kan automatisch moeilijke woorden of vakterminologie extraheren uit een tekst, behalve O4 en O5, die dit wel kunnen met behulp van een expliciete prompt.

Wat betreft samenvattingen kunnen O1, O2, O3, O4 en O5 zowel extraherende als abstraherende samenvattingen maken van de oorspronkelijke tekst. E1, E2 en E3 kunnen alleen een extraherende samenvatting maken, maar dit gebeurt pas na een handmatige selectie van de zinnen.

Alleen O4 en O5 hebben de mogelijkheid om een gegeven tekst te herschrijven. Dezelfde prompt kan leiden tot een ontelbaar aantal resultaten. Alle andere toepassingen slagen hier niet in en alle versies van een herschreven artikel leiden tot hetzelfde resultaat. Tot slot kunnen enkel O1, O2, O4 en O5 wel onregelmatige werkwoorden corrigeren.

Wont-haves

Op het vlak van toegankelijkheid beschikken O4 en O5 over een mobiele versie. Daarnaast kan een gebruiker O1, O2 en O3 wel via een mobiel apparaat bekijken, maar deze lenen geen speciale interface toe. Vervolgens bieden E1, E2 en E3 geen mobiele versie aan.

Daarnaast beschikken enkel E1, E2 en E3 over luistersoftware. Hoewel browsers over ingebouwde luistertools beschikken, toch bieden de andere tools geen personaliseerbare *text-to-speech* techniek aan. Tot slot beschikken de geteste toepassingen over geen integratie met andere spelcheckers. Wel werkt de browserextensie van *Grammarly* bij zowel O1, O2, O3, O4 en O5.

4.2. Geschikte taalmodel voor gepersonaliseerde tekstvereenvoudiging met ATS

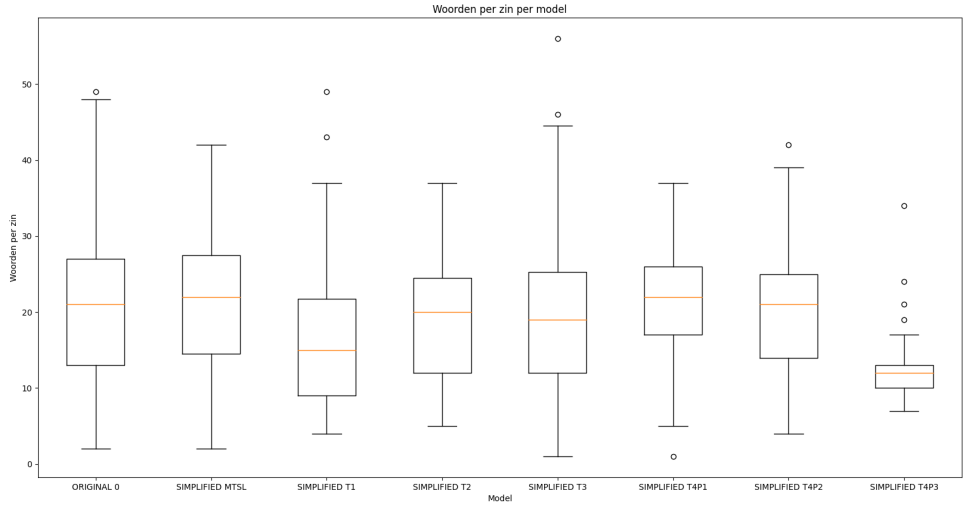
De vergelijkende studie evalueert de uitvoer van de uitgeteste taallmodellen, opgesomd in 3.6, met een machinale en een menselijke beoordeling. Zo achterhaalt deze onderzoeksmethode welk taalmodel of LLM beter aansluit bij het aanbieden van gepersonaliseerde ATS voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

Machinale beoordeling van de vereenvoudigde teksten

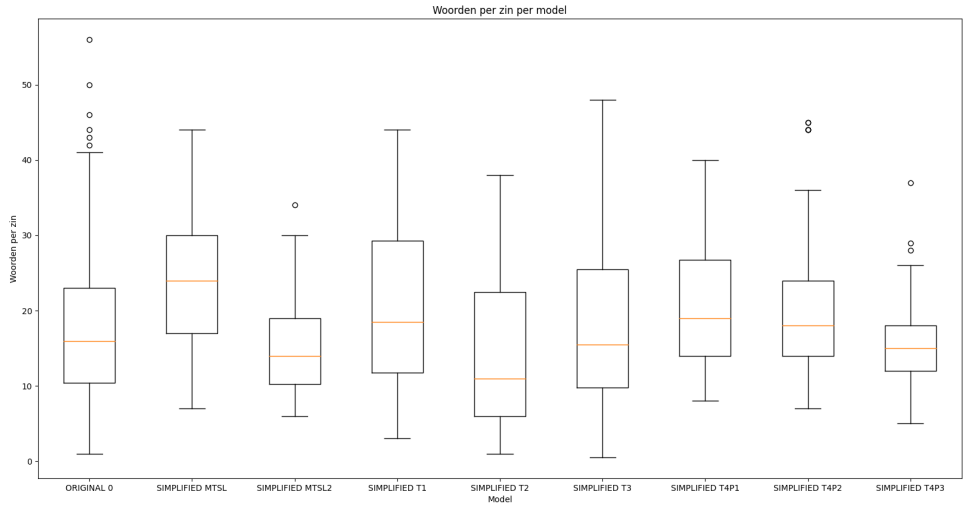
Tabel 4.2 geeft het aantal zinnen per (vereenvoudigd) artikel. De MTS-referentieteksten bevatten minder zinnen dan het oorspronkelijk artikel. Het aantal zinnen na ATS met T1, T2 en T3 is gehalveerd tot minder dan een kwart van oorspronkelijke hoeveelheid zinnen. Enkel T4P2 genereert meer zinnen dan de oorspronkelijke versie van A1 na ATS. T4P2 genereert bij zowel A1 als A2 meer zinnen vergeleken met de andere geteste taalmodellen. T2 daarentegen genereert bij beide artikelen het minst aantal zinnen. Figuren 4.4 en 4.5 illustreren deze verschillen tussen de taalmodellen.

Bron	Zinnen in A1	Zinnen in A2
Oorspronkelijk	78	159
MTS (door leerkracht)	43	45
MTS (door leerling)	n.v.t.	50
T1	26	24
T2	11	7
T3	67	130
T4 P1	61	98
T4 P2	89	133
T4 P3	39	55

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



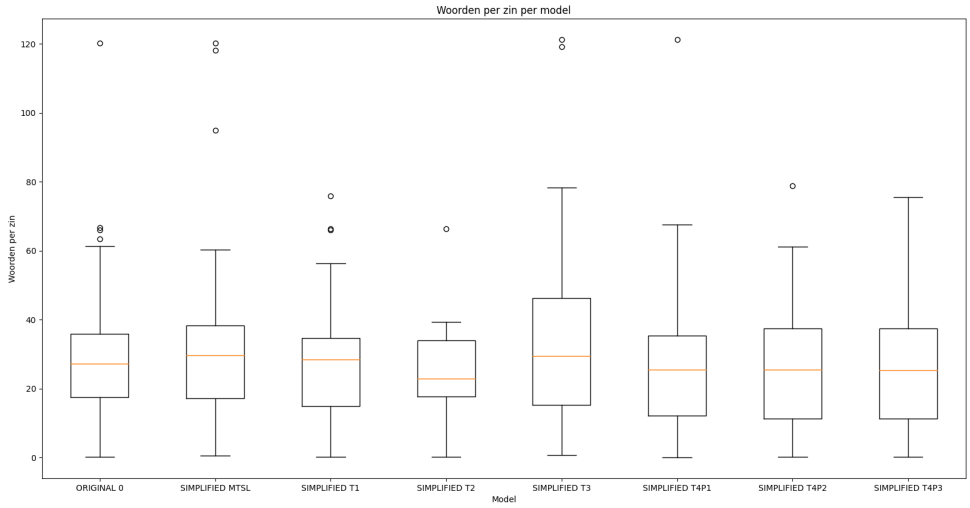
Figuur (4.4)
Overzicht van het minimum, maximum en gemiddeld aantal woorden per zin per model in A1.



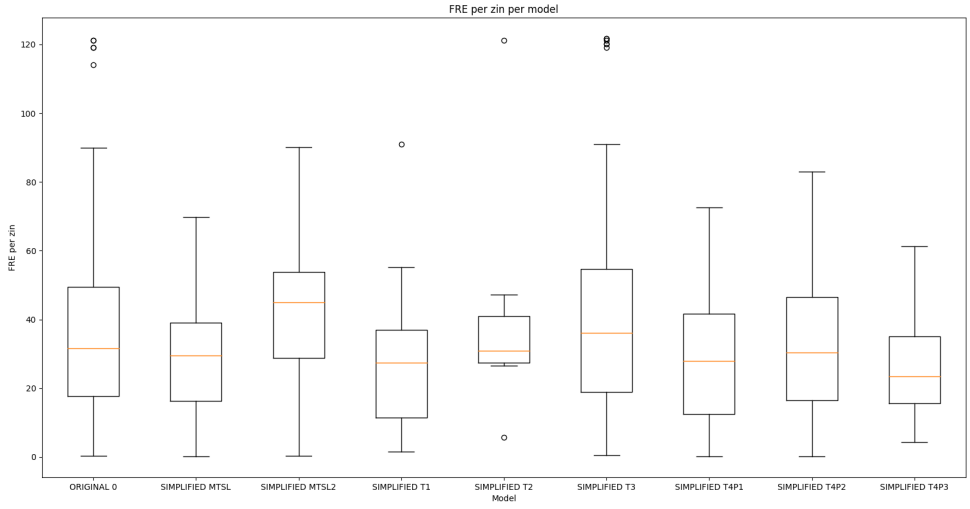
Figuur (4.5)
Overzicht van het minimum, maximum en gemiddeld aantal woorden per zin per model in A2.

Verder vergelijkt het onderzoek de verkregen leesbaarheidsscores per zin dat ieder taalmodel kan genereren. Allereerst de FRE-scores die de leesgraad van een zin aanduidt. Alle geteste taalmodellen genereren zinnen waarvan de FRE-scores niet opmerkelijk hoger of lager zijn ten opzichte van het oorspronkelijk artikel. Figuren 4.6 en 4.7 tonen deze verschillen. Gemiddeld bevinden alle versies van het wetenschappelijk artikel zich tussen 20 en 50. Zonder *outliers* beperkt T3 de FRE van alle zinnen tot hoogstens 40. T3, T4P1, T4P2 en T4P3 genereren zinnen met een hogere

FRE dan OG en MTSL.



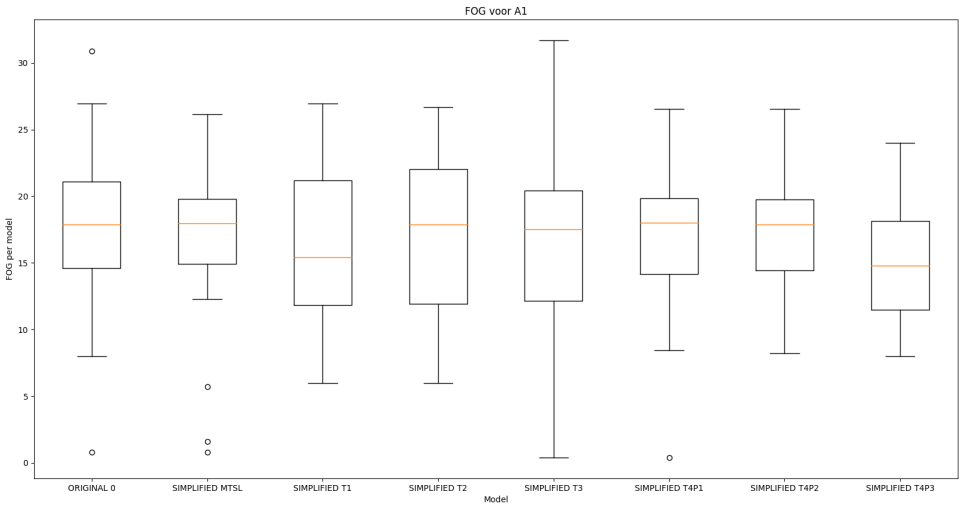
Figuur (4.6)
Boxplot van de FRE-scores voor A1.



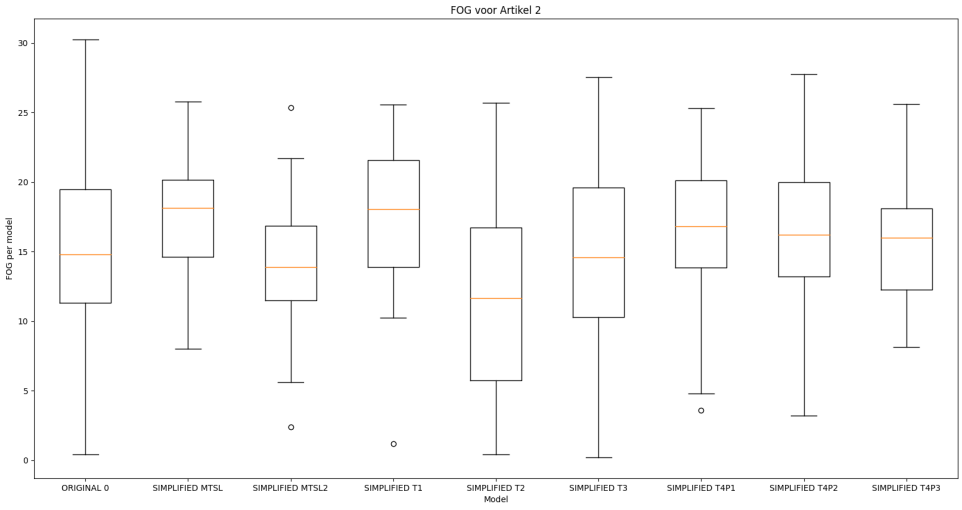
Figuur (4.7)
Boxplot van de FRE-scores voor A2.

De FOG-scores van alle geteste taalmodellen en MTS-referentieteksten zijn niet opmerkelijk hoger of lager bij de vereenvoudigde wetenschappelijke artikelen, zoals weergegeven in figuren 4.8 en 4.9. De zinnen van MTSL2 en T2 scoren gemiddeld lagere FOG-scores dan OG. Daarnaast scoort T2 een lager gemiddelde dan andere taalmodellen. Dit gemiddelde ligt tussen 10 en 12. Tot slot scoren MTSL en andere taalmodellen gemiddeld hoger dan OG. Tot slot genereren de taalmodellen geen

zinnen met een hogere FOG-score dan OG.



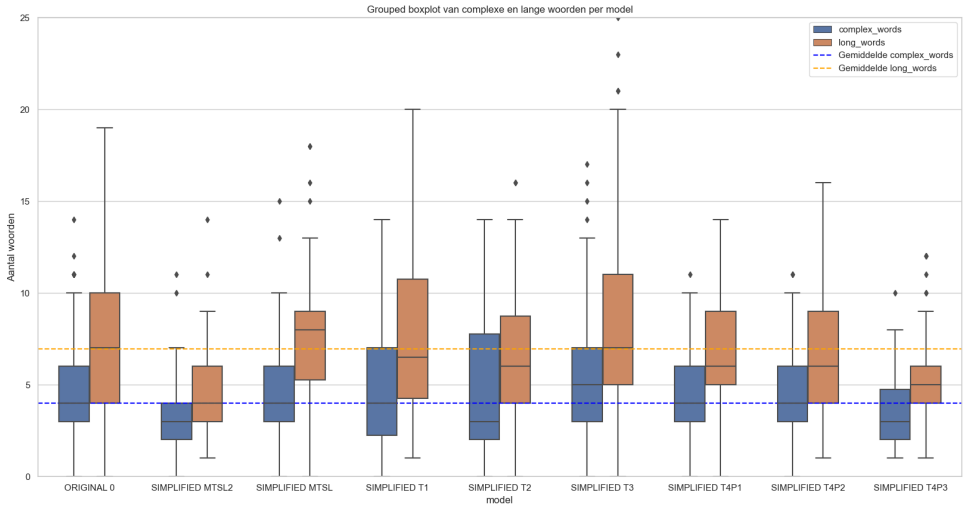
Figuur (4.8)
Boxplot van de FOG-scores voor A1.



Figuur (4.9)
Boxplot van de FOG-scores voor A2.

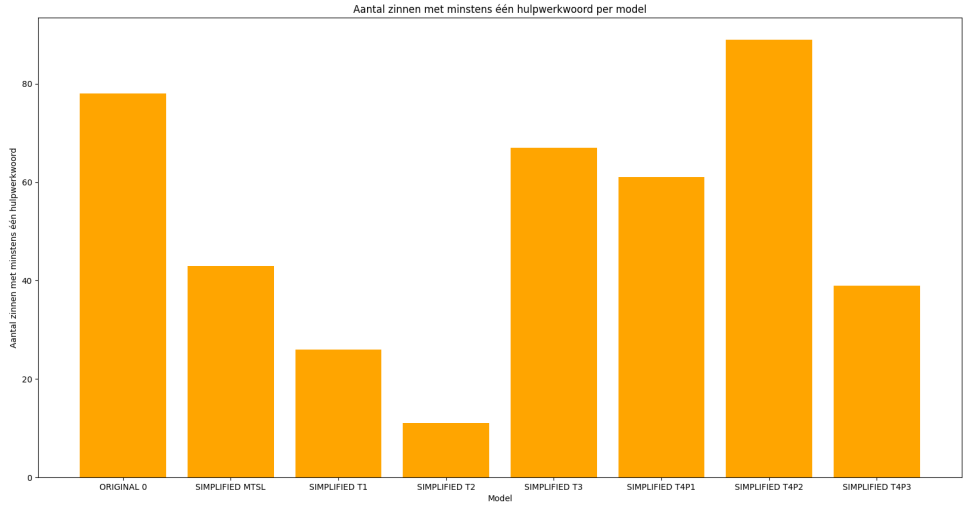
Vervolgens meet het onderzoek het aantal lange en complexe woorden die de taalmodellen genereren. Deze resultaten geeft figuur 4.10 weer. Zo genereren T1, T2 en T3 meer complexe woorden vergeleken met T4, MTSL en OG. Bij A1 genereert T4P3 opmerkelijk minder complexe woorden per zin dan de andere taalmodellen. Verder volgt A2 een gelijke trend met de andere taalmodellen. Het gebruikte script ziet een woord als 'lang' wanneer dit minstens vier lettergrepen heeft. Tot slot ge-

nereren MTSL, T2, T4P1, T4P2 en T4P3 minder lange woorden per zin dan OG.

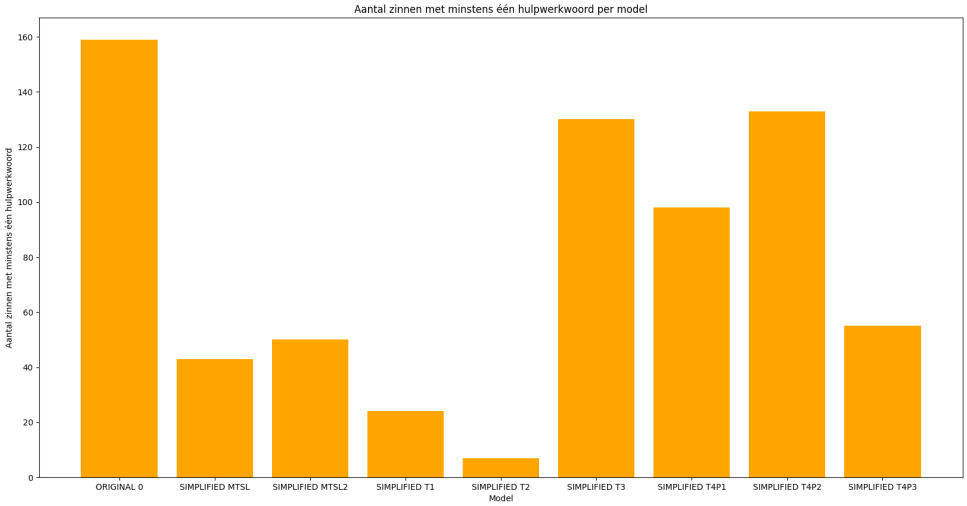


Figuur (4.10)
Een boxplot van het aantal lange en complexe woorden per zin, gegroepeerd op model voor A1.

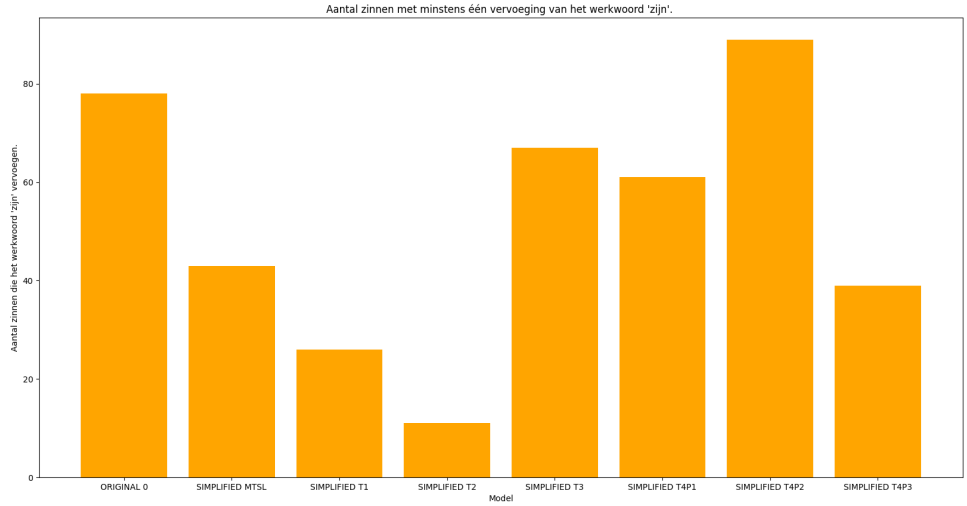
Vervolgens tonen figuren 4.11 en 4.12 het aantal hulpwerkwoorden in de tekst. Deze figuren zijn geen absolute percentages en houden geen rekening met het aantal zinnen. Ten slotte tonen 4.11 en 4.12 het aantal vervoegingen van het werkwoord 'zijn' aan.



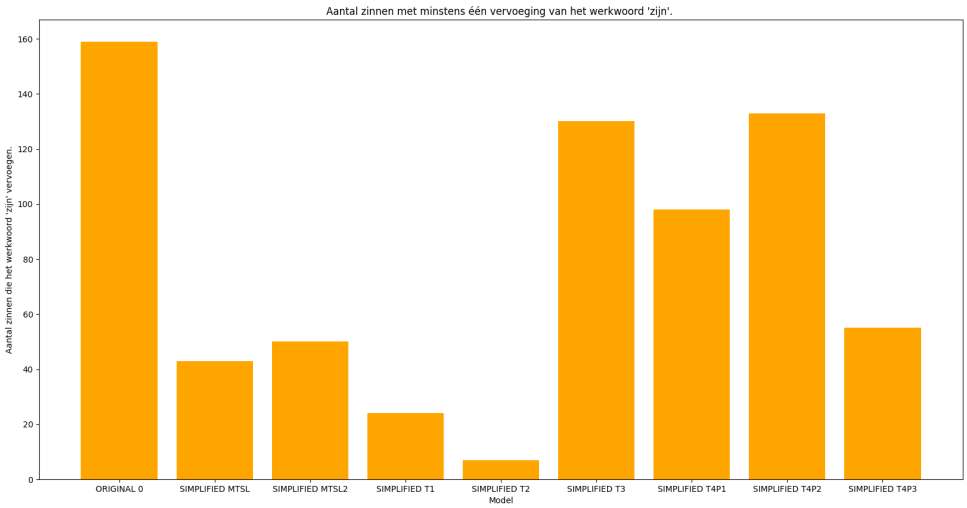
Figuur (4.11)
Een staafdiagram van het aantal gebruikte hulpwerkwoorden in de tekst, gegroepeerd op model voor A1.



Figuur (4.12)
Een staafdiagram van het aantal gebruikte hulpwerkwoorden in de tekst, gegroepeerd op model voor A2.



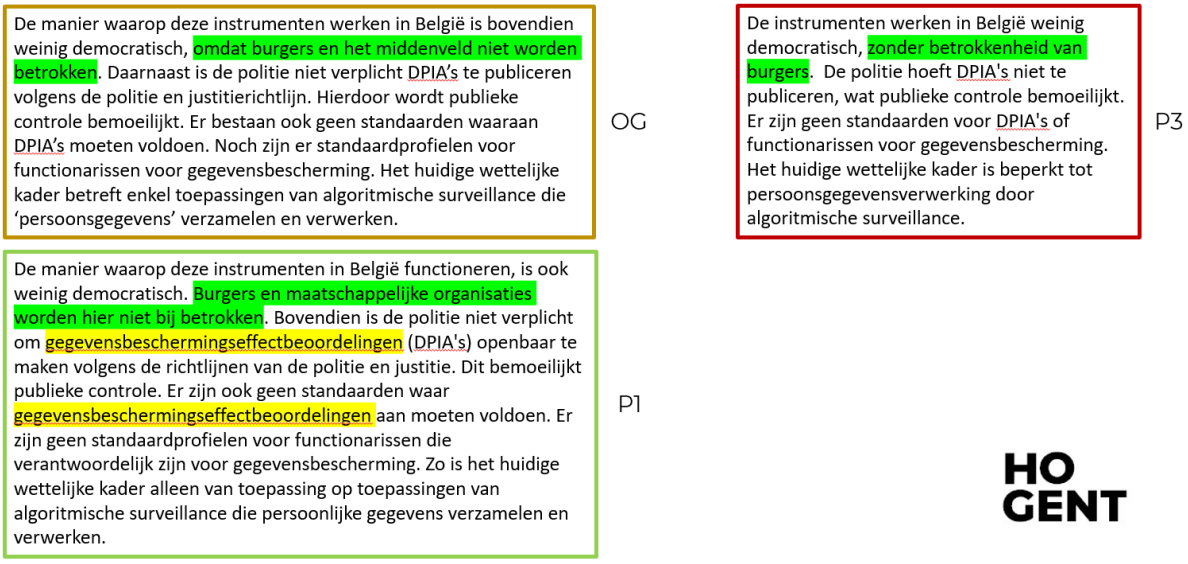
Figuur (4.13)
Het aantal vervoegingen van het werkwoord ’zijn’, gegroepeerd op model voor A1.



Figuur (4.14)
Het aantal vervoegingen van het werkwoord 'zijn', gegroepeerd op model voor A2.

Menselijke beoordeling van de referentieteksten.

In het volgende deel bespreekt het onderzoek de menselijke beoordeling van de resultaten. Allereerst kunnen T4P1 en T4P2 Engelstalige vaktermen vertalen 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. De auteurs schrijven alle afkortingen voluit, zoals beschreven in de richtlijnen. Zo toont figuur 4.15 deze verschillen.



Figuur (4.15)
De verschillen tussen de oorspronkelijke tekst, T4P1 en T4P3 bij één uitgekozen paragraaf.

Alle taalmodellen kunnen LS 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. T4P3 verkort lange zinnen door deze op te splitsen. T1, T2 en T3 behalen een gelijke zinslengte als dat van de oorspronkelijke zin. T4P1 en T4P2 kunnen langere zinnen genereren, maar smelten geen twee zinnen met elkaar samen.

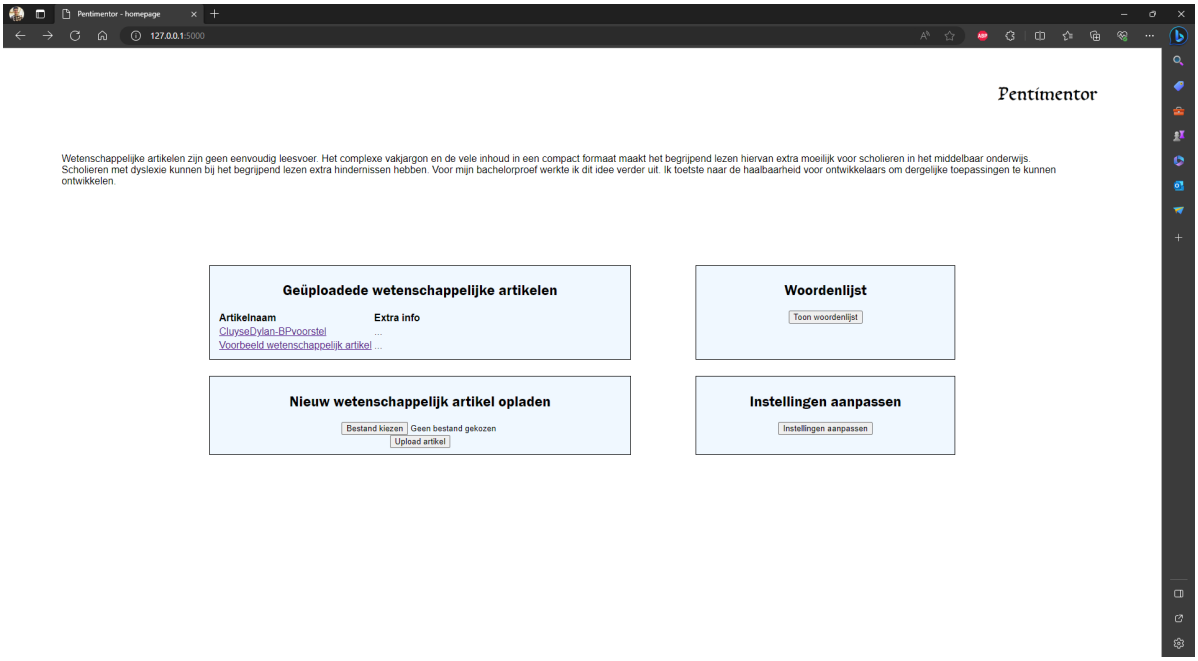
Geen taalmodel wijkt af van de hoofdgedachte van het oorspronkelijke wetenschappelijk artikel. Hoewel T1, T2 en T3 deels afgebroken zinnen kan genereren, bevatten deze zinnen de hoofdgedachte. T2 bevat minder dan 10% van het oorspronkelijk artikel en ontbreekt daarbij bijzaken die nodig zijn om alle vragen in B te kunnen begrijpen en te beantwoorden. 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.

Ter conclusie van de resultaten scoren de drie prompts van T4 beter bij de menselijke beoordeling van de resultaten. Het taalmodel en de verwante drie prompts genereren coherente teksten met een verlaagde lexicale complexiteit. Echter houden de geteste taalmodellen weinig tot geen rekening met afkortingen of bronvermeldingen.

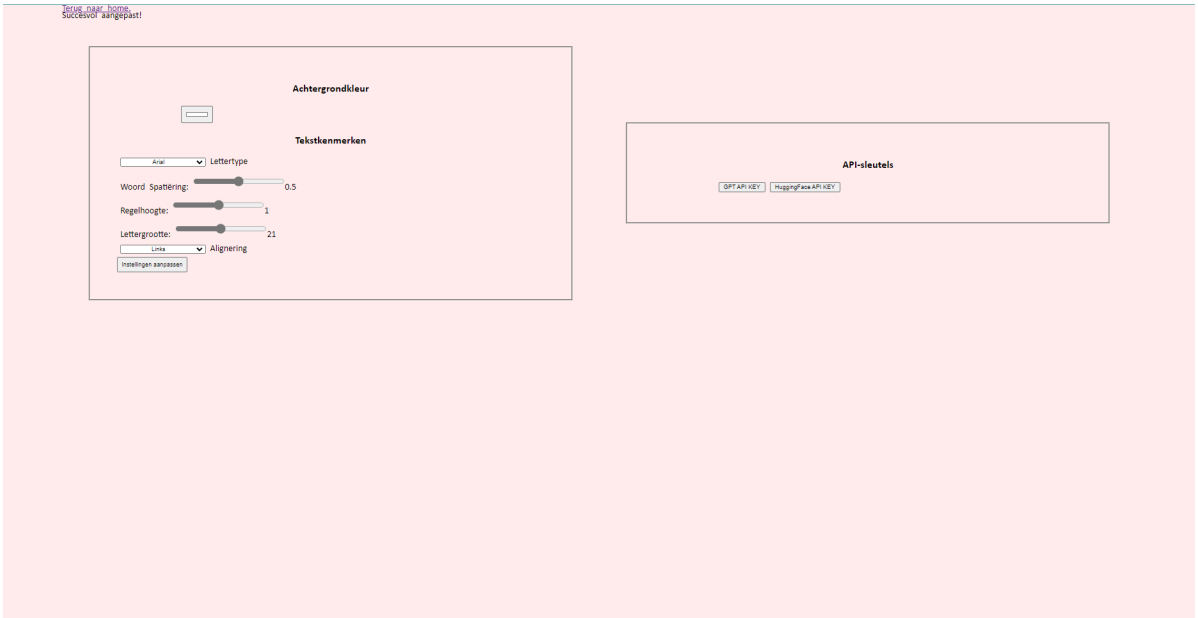
4.3. Pentimentor vergelijken met *top-of-the-line* tools.

In de volgende sectie bespreekt het onderzoek het bekomen prototype. Dit prototype kan de lezer terugvinden op de volgende Github-repository[2]. Gebruikers kunnen vanuit de homepagina uit de verschillende functionaliteiten kiezen, namelijk het uploaden van een wetenschappelijk artikel, het tonen van een geüpload wetenschappelijk artikel, opmaakopties toevoegen en de woordenlijst tonen. Op de instellingenpagina kunnen eindgebruikers hun persoonlijke opmaak toevoegen. Zo toont figuur 4.17 alle mogelijke opmaakopties die Pentimentor aanreikt. Pentimentor past deze aanpassingen toe op alle webpagina's.

[2]https://github.com/Dyashen/pentimentor/tree/main



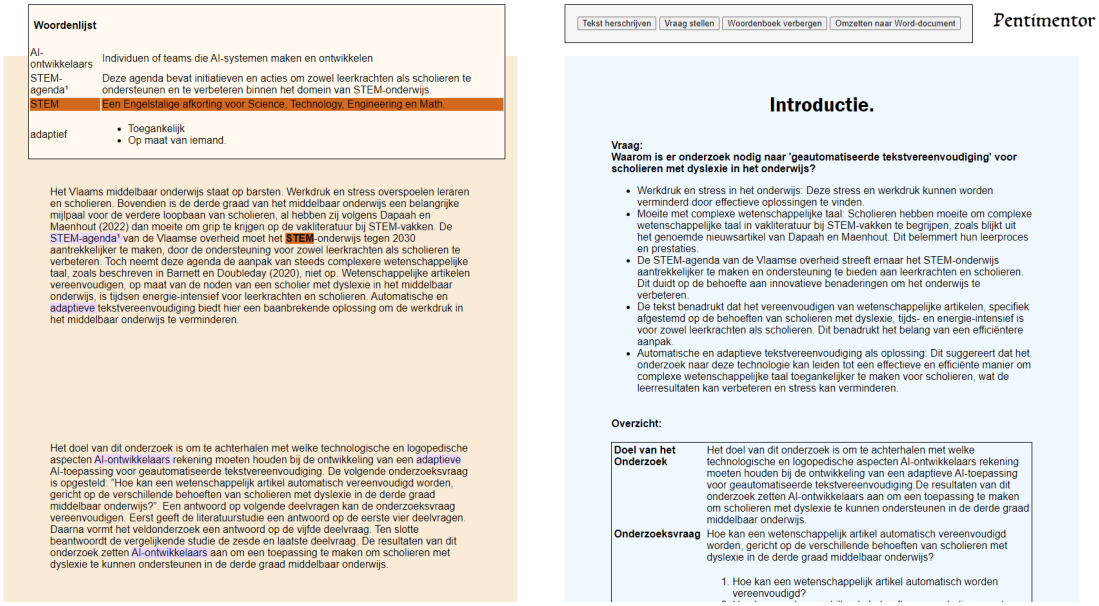
Figuur (4.16)
Voorbeeldweergave van de homepagina.



Figuur (4.17)
Voorbeeldweergave van de instellingenpagina.

Vervolgens toont figuur 4.18 een mogelijke weergave van het scholierencomponent. In deze weergave kunnen scholieren een zelfgekozen tekst markeren om

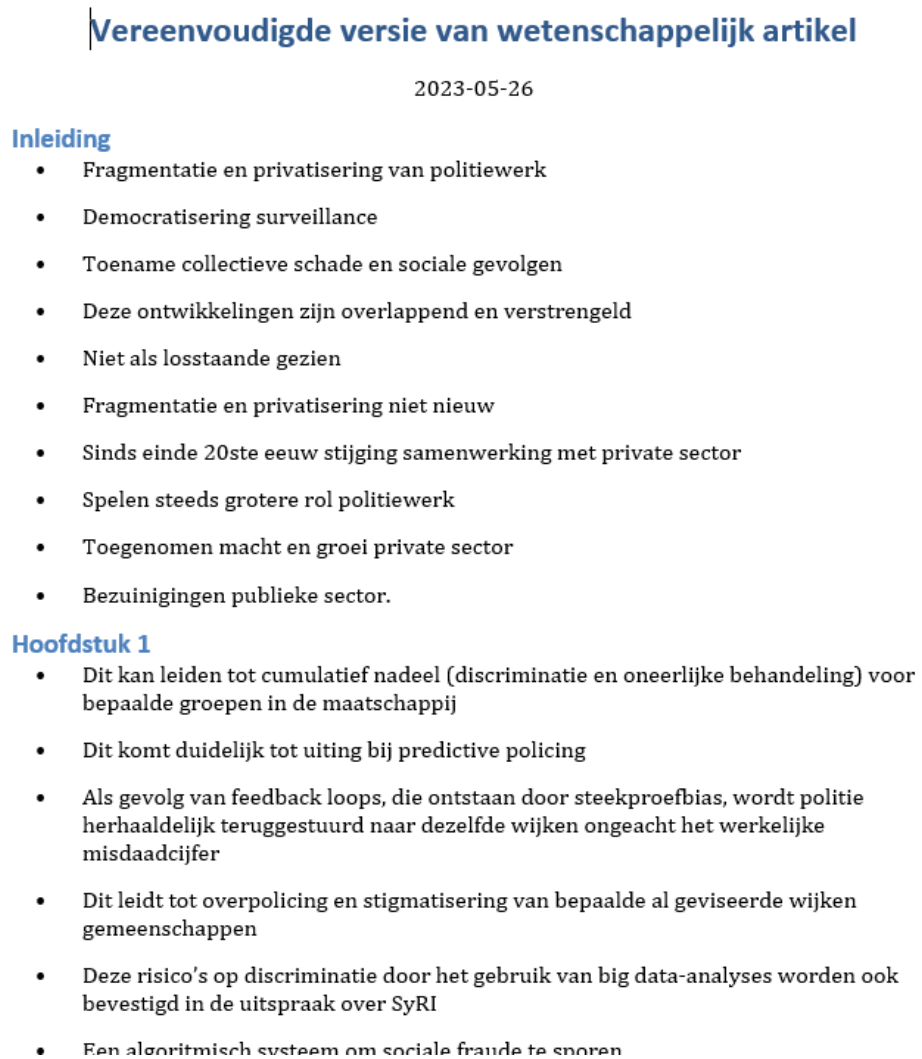
deze te laten vereenvoudigen met gepersonaliseerde ATS. Eerst toont de schermafbeelding hoe gebruikers met Pentimentor de gemarkeerde doorlopende tekst kunnen laten herschrijven naar een tabel. Eerst markeren zij een stuk tekst om hiermee opties aan het mee te geven. Tot slot toont de schermafbeelding hoe scholieren een specifieke vraag kunnen stellen. Eerst geven zij een vraag in met een gecentreerd invoerscherm. Daarna zien ze de uitvoer naast de gemarkeerde tekst. Bovendien stelt Pentimentor gebruikers in staat om op basis van gekregen parameters automatisch personaliseerbare docx-documenten te genereren. Dit gebeurt automatisch op maat van de instellingen die zij op de instellingenpagina doorvoeren.



Figuur (4.18)
Voorbeeldweergave van het scholierencomponent.

Volgens de evaluatie en experimenten blijkt Pentimentor te voldoen aan de *must-haves* uit de requirementsanalyse. Tabel 3.4 geeft hier een overzicht van. Het biedt twee methoden aan om pdf-bestanden in te lezen. Dit via een pipeline van machineleer- en OCR-technieken. Bovendien kan het de tekst ophalen met behulp van PDFMiner, wat overeenkomt met de verwachte functionaliteiten voor pdf-upload. Daarnaast hebben gebruikers de vrijheid om te kiezen welke tekstinhoud ze willen vereenvoudigen met behulp van gepersonaliseerde ATS. Na de vereenvoudiging of samenvatting van een wetenschappelijk artikel kunnen eindgebruikers alle vragen beantwoorden met behulp van de inhoud van het vereenvoudigde artikel, zoals aangegeven door Hollenkamp (2020) als een absolute vereiste.

Verder kunnen gebruikers de opmaak van Pentimentor aanpassen naargelang hun voorkeur. Zo past het systeem deze voorkeuren toe op de digitale weergave in de webtool, maar ook de opmaak van het uitvoerbestand. Daarnaast houdt Pentimentor rekening met de gekozen regeleindes, woord- en karakterspatiëring, lettertype -en grootte, koppenstructuur en marges van het uitvoerbestand. Pentimentor houdt hier rekening mee, in tegenstelling tot de andere uitgeteste toools. Enkel E1 en E2 kunnen het lettertype -en grootte aanpassen. Tot slot toont figuur 4.19 hoe een volledig personaliseerbaar docx-bestand er uit kan zien.



Vereenvoudigde versie van wetenschappelijk artikel

2023-05-26

Inleiding

- Fragmentatie en privatisering van politiewerk
- Democratisering surveillance
- Toename collectieve schade en sociale gevolgen
- Deze ontwikkelingen zijn overlappend en verstrengeld
- Niet als losstaande gezien
- Fragmentatie en privatisering niet nieuw
- Sinds einde 20ste eeuw stijging samenwerking met private sector
- Spelen steeds grotere rol politiewerk
- Toegenomen macht en groei private sector
- Bezuinigingen publieke sector.

Hoofdstuk 1

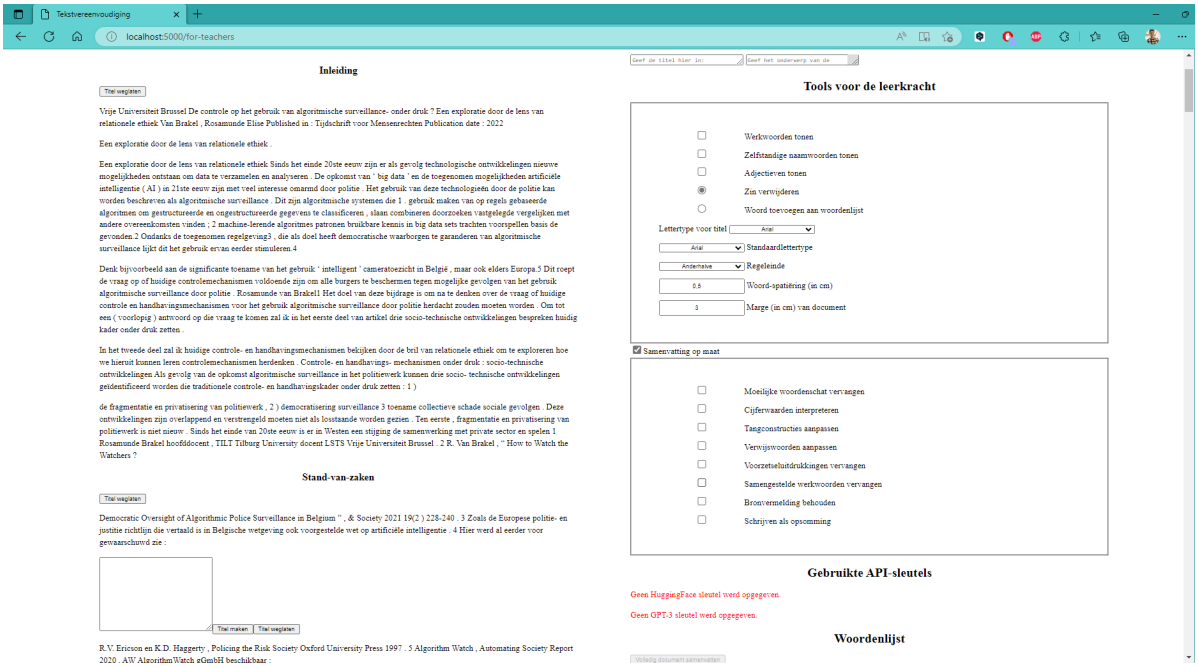
- Dit kan leiden tot cumulatief nadeel (discriminatie en oneerlijke behandeling) voor bepaalde groepen in de maatschappij
- Dit komt duidelijk tot uiting bij predictive policing
- Als gevolg van feedback loops, die ontstaan door steekproefbias, wordt politie herhaaldelijk teruggestuurd naar dezelfde wijken ongeacht het werkelijke misdaadcijfer
- Dit leidt tot overpolicing en stigmatisering van bepaalde al geviseerde wijken gemeenschappen
- Deze risico's op discriminatie door het gebruik van big data-analyses worden ook bevestigd in de uitspraak over SyRI
- Een algoritmisch systeem om sociale fraude te sporen

Figuur (4.19)
De uitvoer na een vereenvoudiging met Pentimentor. De tekst is een vereenvoudigde versie van het artikel van Van Brakel (2022).

Verder bevat Pentimentor enkele *should-haves*. Allereerst kunnen gebruikers een

tekst op een duidelijke manier markeren. Hiermee kunnen zij annotaties toevoegen, de tekst aanpassen door *in-line* definities toe te voegen. Bovendien kan Pentimentor abstraherende samenvattingen genereren in verschillende formaten, zoals opsommingen, tabellen of doorlopende tekst. Zo toont figuur 4.19 een voorbeeld van een gegenereerde opsommingssamenvatting.

Pentimentor zorgt voor een duidelijke gebruikerservaring door meldingsschermen te tonen wanneer het iets van de gebruiker verwacht. Zo toont het onder andere waarschuwingen in formulieren, zoals getoond in figuur 4.20.



Figuur (4.20)
Een mogelijke weergave van het lerarencomponent met het wetenschappelijk artikel van Van Brakel (2022) als input.

Echter voldoet Pentimentor niet aan alle *should-haves*. Zo ontbreekt het de mogelijkheid om automatisch een woordenlijst met moeilijke woorden of vakjargon te genereren. Daarnaast mist Pentimentor analytische functionaliteiten, zoals het tonen van tekstanalyse aan de eindgebruiker.

Tot slot bevat Pentimentor geen *wont-haves*. Zo ontbreekt het een luistercomponent waarmee scholieren de vereenvoudigde tekst kunnen beluisteren. Deze functionaliteit is wel aanwezig bij E1, E2 en E3. Andere uitgeteste tools beschikken hier ook niet over. Bovendien kunnen gebruikers Pentimentor alleen raadplegen in een lokale omgeving met Docker, terwijl zij wel andere geteste toepassingen zonder installatie kunnen raadplegen. Daarnaast heeft Pentimentor geen browserextensie, terwijl O5 dit als enige toepassing wel kan.

5

Conclusie

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

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

Eerst geeft de requirementsanalyse nieuwe inzichten in huidige toepassingen voor *automatic text simplification* (ATS). Zo ontbreekt er personaliseerbaarheid in bestaande online tools. Verder beschikken tools die enkel lexicale vereenvoudiging toepassen over onvoldoende opmaakopties om de leeservaring van scholieren met dyslexie tijdens het begrijpend lezen van een wetenschappelijk artikel te bevorderen. Daarnaast kunnen eindgebruikers met deze toepassingen geen gepersonaliseerde vereenvoudiging of samenvatting maken. Ten slotte kunnen zij geen wetenschappelijke artikelen inladen in de toepassingen die wel gepersonaliseerde ATS kunnen uitvoeren.

Bing Chat en ChatGPT bieden mogelijkheden voor ATS aan. Ze vereisen echter een uitgebreide informaticakennis, ofwel een vaardigheid waarover de meeste scholieren en leraren niet beschikken. Ontwikkelaars kunnen de achterliggende taalmodellen gebruiken om toepassingen te maken, maar zij richten zich hoofdzakelijk op samenvattingstools. In hoofdzaak resulteert deze bijdrage niet in meer begrijpelijke teksten. Het komt het leerproces van scholieren vaak niet ten goede. Huidige toepassingen bewijzen nochtans dat ontwikkelaars toepassingen voor personaliseerbare tekstvereenvoudiging kunnen ontwikkelen. De opgestelde requirementsanalyse benadrukt de noodzaak van een gebruiksvriendelijke toepassing in het onderwijs, waarmee scholieren en leerkrachten wetenschappelijke teksten op een efficiënte manier kunnen vereenvoudigen.

Vervolgens wijst de vergelijking van taalmodellen uit dat HuggingFace (HF) taalmodellen, specifiek getraind op vereenvoudigingsopdrachten, lexicale vereenvoudiging mogelijk maken. Het geavanceerde taalmodel GPT-3 doet het beter door bovendien syntactische vereenvoudiging aan te bieden, samen met formaatwijzigingen. Dit is ongezien bij huidige toepassingen. Zo produceert GPT-3 ook teksten met minder lange en complexe woorden. Dit taalmodel kan doelgroepen in grote lijn inschatten, waartoe andere tools niet in staat zijn. Geteste HF taalmodellen genereren minder coherente teksten en lopen het risico op samengesmolten zinnen. Daarnaast vereisen zij een extra vertaalfase, wat GPT-3 niet hoeft te doen. Zo moeten ontwikkelaars geen extra vertaalfase uitwerken wanneer zij teksten met GPT-3 willen vereenvoudigen. Daarbij moet het prototype specifieke prompts en technieken aangegeven door McFarland (2023) en White e.a. (2023) gebruiken.

Tot slot toont de vorming van Pentimentor aan dat ontwikkelaars ATS-software kunnen ontwikkelen met *open-source* AI- en NLP-technologieën. Zo kunnen zij PDFMiner en Layoutparser gebruiken om tekstinhoud uit wetenschappelijke artikelen te extraheren, met of zonder behoud van de oorspronkelijke titelstructuur. Bovendien kunnen ze de API van OpenAI's GPT-4 benutten voor gepersonaliseerde ATS-toepassingen door middel van geschikte prompts. Vervolgens kunnen zij met Pandoc gepersonaliseerde documenten in docx-formaat automatisch genereren. Ontwikkelaars kunnen basis Javascript toepassen om eenduidige handelingen voor eindgebruikers te onwikkelen, die voordien enkel per commandline mogelijk waren. Zo kunnen zij webpagina's opbouwen die voldoen aan de noden beschreven in Rello e.a. (2012a). Hoewel het prototype niet aan alle *should-haves* en *could-haves* voldoet, kunnen ontwikkelaars een volledig functionele toepassing uitwerken met de genoemde softwarepakketten.

Dit onderzoek legt de nadruk op de aanwezigheid van geavanceerde taalmodellen en tools voor potentiële ATS-toepassingen die voldoen aan de behoeften van scholieren met dyslexie. GPT-3 kan dienen als een geschikt taalmodel, vanwege zijn sterke prestaties in toegepaste leesmetrieken, waaronder criteria zoals het aantal complexe en lange woorden per zin. Daarnaast kan dit taalmodel korte annotaties genereren voor vakjargon of onbekende woordenschat voor scholieren. Verder hebben ontwikkelaars de mogelijkheid om zich te richten op specifieke doelgroepen via aanpasbare parameters. Hierdoor kunnen teksten op maat worden gemaakt die aansluiten bij de individuele behoeften van de gebruiker. Ontwikkelaars moeten echter voorzichtig zijn met dergelijke toepassingen, omdat taalmodellen geen gegarandeerd correcte inschatting bieden voor de doelgroepen. Extra trainingsdata kan het model helpen bij deze inschatting door middel van leerstof op leesniveau van de doelgroep, zoals aangeraden door Gooding (2022).

6

Discussie

Om te achterhalen hoe ontwikkelaars gepersonaliseerde *automatic text simplification* (ATS) kunnen bieden, gebruikt het onderzoek drie onderzoeksmethoden. Het onderzoek focust enkel op gepersonaliseerde ATS voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

Allereerst wijzen de resultaten van de requirementsanalyse uit dat zowel erkende toepassingen voor tekstvereenvoudiging als online tools onvoldoende functionaliteiten bieden voor personalisatie naar specifieke behoeften. Dit resultaat komt overeen met de verwachting dat ontwikkelaars hun toepassingen niet richten op gepersonaliseerde ATS. Daarnaast komt het overeen met de verwachtingen dat ontwikkelaars geen rekening houden met dyslectische scholieren in de derde graad van het middelbaar onderwijs. Hiervoor geeft het onderzoek enkele verklaringen: de complexiteit die gepaard gaat met de ontwikkeling van zulke toepassingen, het gebrek aan initiatief binnen het informatica vakgebied en de beperkte populariteit van samenvattingstools, zoals aangegeven door Gooding (2022) in de literatuurstudie.

Hoewel de functionaliteiten van ChatGPT en Bing Chat mogelijk kunnen bijdragen aan gepersonaliseerde ATS, ontbreken deze toepassingen eenduidige handelingen. Dit vormt een obstakel voor gebruikers waardoor zij moeite kunnen ervaren bij het handmatig vereenvoudigen van wetenschappelijke artikelen. Daarnaast houdt ChatGPT geen rekening met verwijzingen of artikelen buiten de getrainde data, wat de credibiliteit van de teruggekregen tekst kan verlagen. Bing Chat vermijdt dit probleem door rekening te houden met externe referenties. Daarmee geeft het systeem, naast de vereenvoudigde tekst, ook direct de aangehaalde referenties mee aan de eindgebruiker. Ontwikkelaars kunnen deze toepassing of dergelijke API gebruiken om nauwkeurige toepassingen met referentiemateriaal aan

te bieden in ondersteunende onderwijstoepassingen. Verder onderzoek naar de toepassing van deze AI-technologieën via een API is noodzakelijk en kan baanbrekend zijn voor de onderwijssector, ondersteund door Garg (2022) en Roose (2023). Anderzijds 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. Tot slot moeten onderzoekers meer experimenten uitvoeren naar het gebruik van Bing Chat en ChatGPT in het onderwijs.

Bij de tweede onderzoeksmethode vergeleek het onderzoek vier verschillende *pretrained* taalmodellen. Deze onderzoeksfase wijst uit dat ontwikkelaars het GPT-3 taalmodel kunnen gebruiken voor gepersonaliseerde ATS. Het onderzoek gaat niet verder dan het finetunen van de API-parameters. Zo bevat het geen extra tekstdata van wetenschappelijke artikelen. Verder wijst de vergelijkende studie uit dat de drie geteste HuggingFace (HF) taalmodellen en het geteste GPT-3 taalmodel via API beschikken over *Complex Word Identification* (CWI)-functionaliteiten en *substitution generation*. Hoewel de drie HF taalmodellen een gekregen tekst op lexicaal vlak kunnen vereenvoudigen, staan ze in de schaduw van GPT-3. Daarmee kan het GPT-3 taalmodel een baanbrekende oplossing bieden voor gepersonaliseerde ATS van wetenschappelijke artikelen. Het kan snel en efficiënt moeilijke woorden herkennen in lange doorlopende tekst. Tot slot kan het de oorspronkelijke tekst herschrijven als tabel of opsomming, wat huidige toepassingen niet aanbieden.

Dit resultaat bevestigt de verwachting dat GPT-3 alle aspecten van gepersonaliseerde ATS kan aanbieden in tegenstelling tot de geteste HF taalmodellen die ze niet allemaal kunnen realiseren. Het onderzoek geeft de complexiteit van de getrainde data als mogelijke verklaring. Zo trainden de onderzoekers de geteste taalmodellen op wetenschappelijke literatuur. Onderzoekers moeten deze verschillen verder onderzoeken. Zo kunnen onderzoekers deze verschillen beter begrijpen, specifiek binnen de context van wetenschappelijke artikelen. Daarnaast kunnen *Large Language Models* of LLM's ook prompts van ontwikkelaars beantwoorden. Deze taalmodellen kunnen verschillende antwoorden genereren, maar het onderzoek wijst uit dat ontwikkelaars de *temperature* parameter kunnen aanpassen om één antwoord per prompt te verkrijgen. Ondanks dat er momenteel onvoldoende inzicht is, zou verder onderzoek zich moeten richten op de verschillen tussen prompts en hun bijhorende *temperature*.

Hoewel GPT-4 niet tot het onderzoek behoort, kunnen ontwikkelaars hiermee toepassingen maken. Simon (2021) geeft aan dat de verschillen tussen deze groei van het aantal parameters weinig effect hebben op de kwaliteit van de tekstdata. Toch beginnen ontwikkelaars snel de sprong te maken naar de volgende iteratie van

het GPT-taalmodel die ook meer energie van het systeem vereist. Verder onderzoek moet uitwijzen of de sprong van GPT-3 naar GPT-4 al dan niet merkbaar is op taalvlak. Een opvolgend onderzoek met dit taalmodel is vereist om te testen of het taalmodel over voldoende data beschikt om wetenschappelijke artikelen te vereenvoudigen op maat van scholieren met dyslexie in de derde graad van het middelbaar onderwijs.

Vervolgens kunnen ontwikkelaars hun eigen specifieke prompts opstellen voor deze taalmodellen. Hierin kunnen zij ook de doelgroep meegeven en daarmee een extra parameter meegeven. Dit onderzoek bevestigt niet in welke mate de uitvoer van elkaar verschilt. Deze eenduidige oplossing kan echter een revolutionaire oplossing bieden voor gepersonaliseerde ATS of samenvatting. Daarom moeten onderzoekers hier verder onderzoek op uitvoeren. Verder onderzoek naar doelgroepinschattingen via prompts is ook nodig. Daarnaast zou toekomstig onderzoek zich kunnen richten op het potentieel van de combinatie van GPT-3 met *full-text-search*-technologieën.

Verder stelt het onderzoek de gebruikte machinale beoordeling in vraag. Zo resulteerde de door taalmodellen vereenvoudigde teksten een miniem verschil bij de leesgraadscores FRE en FOG. Daarnaast wijst het experiment uit dat de *readability*-library de actieve stem van een zin niet kan achterhalen. Daarom kan het onderzoek geen vaststelling maken of de taalmodellen van passief naar actief kunnen schrijven. Spacy *word embeddings* kunnen een alternatieve techniek aanreiken om hulpwerkwoorden en vervoegingen van het werkwoord 'zijn' te achterhalen. Daarnaast kunnen toepassingen zoals TextInspector meer metrieken dan de uitgeteste leesgraadscores aanbieden. Daarom moeten onderzoekers verdere experimenten uitvoeren naar de bruikbaarheid van deze libraries.

De derde en laatste onderzoeksfase toont aan hoe ontwikkelaars toepassingen voor gepersonaliseerde ATS kunnen ontwikkelen, specifiek op maat voor scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Zo dienen de verworven kennis en aangeleerde tools uit de richtingen Toegepaste Informatica aan Vlaamse hogescholen als startpunten om *Pentimentor* te ontwikkelen. Pentimentor dient slechts als een haalbaarheidstoetsing voor ontwikkelaars. Het is belangrijk voor de lezer om te beseffen dat de ontwikkeling gebaseerd is op onderzochte kenmerken en technieken die de impact van handmatige tekstvereenvoudiging al hebben uitgewezen. Daarnaast gebeurt de ontwikkeling van Pentimentor met het oog op een eenduidige implementatie van de technieken, die voordien enkel beschikbaar waren via CLI. Zo wijst de ontwikkeling van Pentimentor uit dat ontwikkelaars met vrij beschikbare middelen en API's wel dergelijk toepassingen kunnen maken. Dit resultaat komt overeen met de verwachting dat ontwikkelaars over de nodige

tools beschikken. Zo baseert het onderzoek zich op de beschikbaarheid van *open source* tools en python-bibliotheken. Die stellen ontwikkelaars in staat om complexe taken eenvoudig te kunnen reproduceren. Toch moet de lezer ervan bewust gemaakt worden dat het doelpubliek van dyslectische scholieren niet werd opgenomen bij de testfase van Pentimentor. Daarom kan dit onderzoek voornamelijk dienen als een haalbaarheidsmeting voor ontwikkelaars.

Tot slot kunnen onderzoekers uit het logopedisch vakgebied Pentimentor gebruiken om onderzoek te voeren naar het leereffect op leesbegrip bij scholieren met dyslexie in de derde graad van het middelbaar onderwijs. Dit stemt overeen met de implicaties waar Gooding (2022) op wijst in haar onderzoek. Daarnaast moeten onderzoekers uit het onderwijsvakgebied de effecten van dergelijk tools observeren bij leerlingen en leerkrachten in het middelbaar onderwijs. Hoewel het onderzoek naar de bruikbaarheid van deze technologieën in het onderwijs zich in een prille fase bevindt, moeten onderzoekers toch de inzet van deze toepassingen verder onderzoeken. Deze bruikbaarheid kan via een browserextensie, lokale of online toepassing.

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 onderzoeksvraag 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-agenda[1] 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 onderzoeksvraag 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 onderzoeksvraag 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 Resoomer[2], Paraphraser[3] en Prepostseo[4]. 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 tijdrovend is en daarmee tekstvereenvoudiging 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 Hero[5], 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 Readable[6]. 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 routinematig 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 NLTK[7] en Spacy[8]. *Deep Martin*[9] 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 tekstinhoud.
Voor Germaanse talen zijn er enkele datasets en word embeddings beschikbaar die de complexiteit van woorden bijhouden. Zo zijn er in de Duitse taal Klexicon[10] en TextComplexityDE[11]. 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-

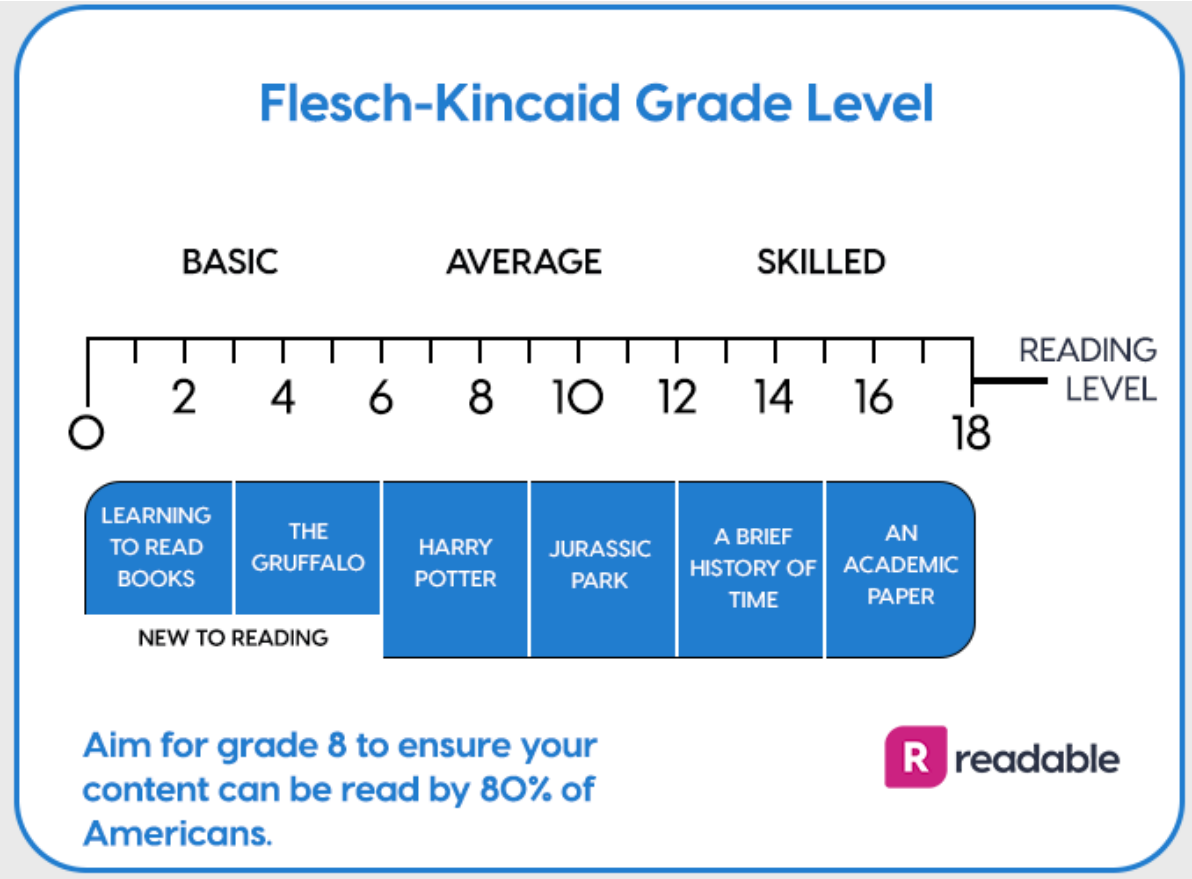
[7] https://www.nltk.org/
[8] https://spacy.io/
[9] https://github.com/chrislemke/deep-martin
[10] https://github.com/dennlinger/klexikon
[11] https://github.com/babaknaderi/TextComplexityDE

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. Figuur A.1 geeft de indeling per doelgroep weer. 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 fases 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 experten 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 thesissen 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 be-

[12] https://pypi.org/project/textstat/



Figuur (A.1)
De indeling van leesgraadscores per doelgroep. Bron: (Readable, 2021)

drijven wordt een longlist opgesteld van beschikbare toepassingen voor tekstvereenvoudiging 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.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.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.3. Structurele aanpassingen

- Het vereenvoudigde artikel volgt dezelfde structuur en chronologische volgorde zoals dat van het oorspronkelijk artikel. Iedere hoofdstuk in het weten-

schappelijk 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.4. Specifieke richtlijnen voor A1

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.5. 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 oliecrisissen 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 internationalisatie 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.

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.

Bosmans, A., Croon, S. & Verreycken, V. (2022a). Woordgebruik - Moeilijke constructies. https://www.vlaanderen.be/taaladvies/taaladviezen/teksten-schrijven/formulering/zinsbouw-moeilijke-constructies

Bosmans, A., Croon, S. & Verreycken, V. (2022b). Woordgebruik - Moeilijke Woorden. https://www.vlaanderen.be/taaladvies/taaladviezen/teksten-schrijven/formulering/woordgebruik-moeilijke-woorden

Bosmans, A., Croon, S. & Verreycken, V. (2022c). Woordgebruik - Synoniemen. https://www.vlaanderen.be/taaladvies/taaladviezen/teksten-schrijven/formulering/woordgebruik-synoniemen

Botelho, F. H. F. (2021). Accessibility to digital technology: Virtual barriers, real opportunities [PMID: 34951832]. *Assistive Technology*, *33*(sup1), 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://clinjournal.org/clinj/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.

Cantos, P. & Almela, Á. (2019). Readability indices for the assessment of textbooks: a feasibility study in the context of EFL. *Vigo International Journal of Applied Linguistics*, 31–52. https://doi.org/10.35869/vial.v0i16.92

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.

Crossley, S. A., Allen, D. & McNamara, D. S. (2012). Text simplification and comprehensible input: A case for an intuitive approach. *Language Teaching Research*, *16*(1), 89–108.

Crossley, S. A., Skalicky, S. & Dascalu, M. (2019). Moving beyond classic readability formulas: new methods and new models. *Journal of Research in Reading*, *42*(3-4), 541–561. https://doi.org/https://doi.org/10.1111/1467-9817.12283

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

Departement onderwijs en vorming. (2023). Voorleessoftware voor Leerlingen met Leesbeperkingen. https://onderwijs.vlaanderen.be/voorleessoftware-voor-leerlingen-met-leesbeperkingen

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). SummEval: 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.slpat-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

Javourey-Drevet, L., Dufau, S., François, T., Gala, N., Ginestié, J. & Ziegler, J. C. (2022). Simplification of literary and scientific texts to improve reading fluency and comprehension in beginning readers of French. *Applied Psycholinguistics*, *43*(2), 485–512. https://doi.org/10.1017/S014271642100062X

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

Lee, J. (2021). Extract text from unsearchable pdfs for data analysis using Python. https://medium.com/social-impact-analytics/extract-text-from-unsearchable-pdfs-for-data-analysis-using-python-a6a2ca0866dd

Leroy, G., Kauchak, D. & Mouradi, O. (2013). A user-study measuring the effects of lexical simplification and coherence enhancement on perceived and actual text difficulty. *International Journal of Medical Informatics*, *82*(8), 717–730. https://doi.org/https://doi.org/10.1016/j.ijmedinf.2013.03.001

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. AI, 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

Matarese, V. (2013). 5 - Using strategic, critical reading of research papers to teach scientific writing: the reading–research–writing continuum. In V. Matarese (Red.), *Supporting Research Writing* (pp. 73–89). Chandos Publishing. https://doi.org/https://doi.org/10.1016/B978-1-84334-666-1.50005-9

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; chatgpt]. 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

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. & A. Baeza-Yates, R. (2015). How to present more readable text for people with dyslexia. *Universal Access in the Information Society*, *16*, 29–49.

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., Bott, S. & Saggion, H. (2013). Simplify or Help? Text Simplification Strategies for People with Dyslexia. *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*. https://doi.org/10.1145/2461121.2461126

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. & 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

Shen, Z., Zhang, R., Dell, M., Lee, B. C. G., Carlson, J. & Li, W. (2021). LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis. *arXiv preprint arXiv:2103.15348*.

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:nbsp; verschillen in economischenbsp; dynamiek en beleid. https://feb.kuleuven.be/research/les/pdf/LES%202022%20-%20197.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