



Kandidatuppsats

IT-forensik och säkerhet 180 hp

Detektering av genererade texter

Stilometri - Är språkmodeller för genomsnittliga?

Digital Forensik 15 hp

Halmstad Jun 16, 2024

Stefan Björk-Olsén



HÖGSKOLAN
I HALMSTAD

Detektering av genererade texter

Stilometri - Är språkmodeller för genomsnittliga?

Stefan Björk-Olsén

Författare	Stefan Björk-Olsén
Program	IT-Forensik och Informationssäkerhet, 180hp
Handledare	Erik Järpe
Examinator	Urban Bilstrup
Lärosäte	Högskolan i Halmstad Högskola
Inlämningsdatum	2024-05-28

Sammanfattning

“Blir inte texter från ChatGPT väldigt genomsnittliga? Hur kommer det sig att ChatGPT räknar fel? Varför har ChatGPT helt rätt i alla stycken förutom kraftiga faktafel i ett av dom?”.

Dessa frågor låg som grund för detta arbete och är sådana som ställts gällande verktyget. Svaren på dessa frågor har hjälpt till att skapa en testmetod för detektering av genererade texter och resultaten från dessa har jämförts med tidigare arbetens resultat. Men detekteringen av genererade texter har visat sig vara problematisk och belyser styrkorna och svagheter med olika metoder för detektering.

Nyckelord: Språkmodell, chattrobot, ChatGPT

Abstract

“Don't texts from ChatGPT become very average? How come ChatGPT miscalculates? Why is ChatGPT factually right in all paragraphs except for major factual errors in one of them?”

These questions were the basis for this work and are ones that has been asked regarding the tool. The answers to these questions have assisted in creating a testing method for the detection of generated texts and the results from these have been compared with the results of previous work. However, the detection of generated texts has proven to be problematic and this work highlights the strengths and weaknesses of different detection methods.

Keywords: Language model, chatbot, ChatGPT

Förord

Författaren önskar tacka alla de som hjälpt till och gjort denna studie till en möjlighet stöttat under processen och kommit med intressanta insikter.

Jag vill också specifikt rikta ett stort tack till med universitetslektor Erik Järpe som gav feedback och en möjlighet till diskussion kring ämnet tidigt under projektets gång. Dels på formalia, ämnet i sin helhet, men även på de statistiska aspekterna av analyserna.

Ordlista

Vissa ord som antingen är allmänt vedertagna eller ord som valts att användas för att beskriva koncept som upprepar sig i text presenteras nedan.

Artificiell Intelligens, AI

Programvara som kan efterlikna naturlig intelligens.

ChatGPT

Är en chatbot som är baserat på en Large Language Model (se ordlistan Large Language Model, LLM), kapabel i att svara på frågor från användaren. ChatGPT använder Open AIs (se ordlistan OpenAI) Large Language Model LM GPT.

Detektor

En programvara vars funktion är att detektera genererade texter. Exempel på detta är GLTR, GPTZero och AI Text Classifier.

False Negative/False Positive

Att ett felaktigt utslag på ett test, där något testar positivt eller negativt på ett test som bör ha gett motsatt utslag.

Genererade Texter

Texter som på något sätt är artificiellt genererade till skillnad från att vara skrivna av en människa. Exempel på detta kan vara svaret som kommer på en given fråga från ChatGPT.

Large Language Model, LLM

En modell tränad på en stor mängd indata för att få ett datorprogram, till exempel en AI, att efterlikna mänskligt språk. På svenska kallad en Språkmodell eller en Stor språkmodell.

Lingvistik

Vetenskapen om det mänskliga språket.

OpenAI

Ett amerikanskt företag som grundades av bland annat Elon Musk för att utveckla Artificiell Intelligens. De kända produkterna från detta företag är Large Language Modell GPT med chatbotten ChatGPT.

Python

Ett programmeringsspråk som ibland beskrivs som ett skriptspråk, det vill säga en typ av programmeringsspråk där källkoden tolkas vid körning, till skillnad från de språk där källkoden tolkas till en kompilerad version av koden och den sen körs.

Språkmodell

En sannolikhetsfördelning för ordföljder.

Stilometri

En metod för statistisk analys av lingvistik där kännetecken i text jämförs med andra texter för att se hur troligt det är att två texter är skrivna av samma person.

Innehållsförteckning

1	Introduktion.....	15
1.1	Bakgrund.....	15
1.2	Frågeställning.....	16
1.2.1	Problemformulering.....	16
1.2.2	Motiverande Aspekter.....	16
1.2.3	Avgränsningar.....	16
2	Metod.....	17
2.1	Val av metoder.....	17
	Litteratursökning.....	17
	Experiment 1 - Simulering.....	18
	Experiment 2 - Prototyper.....	19
2.2	Problematisering.....	20
3	Litteratursökning.....	23
3.1	Tidigare arbeten.....	23
	A thesis that writes itself.....	23
	Detektering av fusk vid användning av AI.....	24
3.2	Språkmodeller.....	26
	Hur genererar en språkmodell sina texter?.....	26
	Artificiell Intelligens (AI).....	26
	Fel i genererade texter.....	26
	Hur detekteras texter i dagsläget?.....	27
3.3	Stilometri.....	27
4	Experiment 1 - Simulering.....	29
4.1	Tester.....	29
5	Experiment 2 - Prototyper.....	31
5.1	Prototyp 1.....	32
	Kravspecifikation.....	32
5.2	Prototyp 2.....	33
	Kravspecifikation.....	33
6	Resultat.....	35
6.1	Test av detektorer.....	35
	Test av GLTR.....	35
	Test av RoBERTa.....	35
	Test av GPTZero.....	36
6.2	Prototyp 1.....	38
6.3	Prototyp 2.....	40
7	Diskussion.....	43
7.1	Framtida arbeten.....	45
8	Slutsatser.....	47
	Referenser.....	49
	Bilaga A - Källkoden Prototyp 1.....	51
	Bilaga B - Källkoden Prototyp 2.....	57

I Introduktion

I.1 Bakgrund

“Blir inte texter från ChatGPT väldigt genomsnittliga? Hur kommer det sig att ChatGPT räknar fel? Varför har ChatGPT helt rätt i alla stycken förutom kraftiga faktafel i ett av dem?”.

Dessa frågor finns som grund för detta arbete och är sådana som ställts gällande verktyget ChatGPT. Med ökande popularitet och användning av sådana verktyg för att generera texter utifrån angivna parametrar, så uppfattas den allmänna kunskapen om dessa verktygs kapacitet som bristfällig [1]. Svaren på dessa frågor är både intressanta men hjälper framförallt till att skapa en modell för att detektera om en text är genererad i ett sådant verktyg.

Inledningsvis kartläggs ämnet för att fastställa hur och varför den genererar den text den gör, och konstatera om det finns någon intelligens i detta verktyg som ofta omnämns som Artificiell Intelligens, AI. En klar bild som utgångspunkt främjar nya synsätt på processen med att utveckla och testa metoder för detektering.

Utöver att kartlägga ämnet och öka förståelsen för den så är målett tvådelat. Den ena delen är att undersöka vilka metoder som finns tillgängliga idag för att detektera genererade texter. Medan den andra relaterade delen är att undersöka om det går att använda sig av statistik och stilometri för att detektera sådana och ställa denna metod i kontrast till de redan tillgängliga. Utöver dessa mål så är syftet att informera om hur en språkmodell såsom ChatGPT från OpenAI [2] fungerar, och därmed även de styrkor och svagheter en sådan har, för att bidra till en ökning av den allmänna kunskapen inom ämnet.

Denna studie startar där det tidigare examensarbetet ”A thesis that writes itself: On the threat of AI-generated essays within academia” [3] slutade. Frågan som ställs är hur situationen ser ut nu två år senare. Har genereringen blivit mer naturlig, har detektorernas träffsäkerhet ökat, eller har kapprustningen mellan generatorerna och detektorerna stagnerat?

Det ställs även frågan om det återfinns en problematik hos de tjänster som idag säger sig kunna detektera genererade texter, och om det går att ta fram en metod baserat på hur språkmodeller är uppbyggda.

1.2 Frågeställning

De tre primära frågeställningarna som ämnas besvaras är:

- Vilka metoder finns idag för att detektera genererade texter?
- Hur står sig dessa metoder mot slumpen eller mänsklig granskning?
- Vilken potential finns det i att detektera genererade texter med hjälp av statistik och stilometri?

1.2.1 Problemformulering

Med första frågan är målsättningen att undersöka vilka metoder som finns publikt tillgängliga för att detektera genererade texter från verktyg såsom ChatGPT.

Den andra frågan är en fördjupning av den första där tillförlitligheten i dessa detektorer undersöks för att fastställa om dessa har en god träffsäkerhet eller är i behov av förbättring.

Till sist hanteras den tredje frågan genom att skapa en metod för detektering och implementera denna i ett programmeringsprojekt. Detta primärt för att se om statistik och stilometri går att använda sig för att detektera genererad text. Sekundärt så undersöks även om sådana metoder har en potential att texter baserat på hur sannolikt det är att de aktuella texterna är genererade eller skrivna av en person.

1.2.2 Motiverande Aspekter

Det som motiverar ett djupare undersökning av ämnet ur ett samhällsperspektiv är dels att informera om begränsningarna med generatorer för texter såsom ChatGPT och även belysa problematiken med att dessa texter är svåra att urskilja från mänskligt skrivna texter.

När det kommer till ett forskningsperspektiv så upplevs området som relativt utforskat, till stor del för att ämnet är så pass nytt. Egna tester och statistisk analys av nya och gamla resultat kommer att tillföra forskning inom området.

1.2.3 Avgränsningar

Fokuset är primärt på detekteringen av genererade texter, det vill säga detektering av om en text är direkt skriven av en faktisk person eller om texten är artificiellt genererad utifrån en språkmodell. Det läggs ingen vikt hur genererade texter används eller dess påverkan på samhället eller akademien.

När det kommer till hur de genererade texterna skapas så har den mest vanligt förekommande valts ut, det vill säga ChatGPT av den senaste publikt tillgängliga versionen, GPT-3.5.

2 Metod

För att på ett strukturerat sätt kunna besvara frågeställningen så har omfattningen delats upp i tre distinkta delar, med tydligt avgränsande metoder, där varje föregående del ligger som grund för följande delar.

Generatoren för text som används är GPT-3.5 via ChatGPT. Detta val av tjänst för generering av texter är baserat på att denna bedöms vara den mest populära i dagsläget.

2.1 Val av metoder

För att kunna svara på frågeställningens frågor så har valet av metoder anpassats med hänsyn till dessa frågor. Men även om metoderna anpassats för att svara på en specifik fråga så används resultaten från samtliga metoder tillsammans för att ge ett mer komplett svar och bakgrund till de val som görs under programmeringen.

Litteratursökning

Den första delen av projektet är en teoretisk del med kartläggning av möjligheterna men även begränsningarna av att använda ChatGPT för att generera texter med hjälp av en språkmodell utifrån givna parametrar. Utöver kartläggningen görs en analys av hur detta verktyg genererar texter, för att dels få en förståelse inför följande delar, men även ge klarhet i varför avvikande egenskaper såsom faktafel eller felberäkningar kan förekomma.

Tidigare arbeten inom området lyfts upp och dess metoder samt resultat ställs emot de metoder och resultat som framkommer här. Redan existerande sätt att detektera genererade texter kartläggs för att sedan användas under testerna.

Denna delen undersöker även om det är korrekt att ge genererade texter stämpeln AI, eller om denna stämpel kan ge en inkorrekt bild av vad språkmodeller är kapabla till. Syftet med detta är dels att få en djupare förståelse för ämnet men även bistå med kunskap till skapandet av den egna detektorn. Frågan som ställs är hur bred den kreativa variationen i de genererade texterna är. Det vill säga hur stor förekomsten av variation utanför angivna parametrar är, vilket kan ha en inverkan på hur användbart Stilometri kan vara på att detektera de genererade texterna.

Kartläggningen fastställer även vilka detektorer för att detektera genererade texter som ska användas baserat på tidigare arbeten.

Det är framförallt i detta skede den första frågan i frågeställningen besvaras, det vill säga vilka sätt det finns i dagsläget för att detektera genererade texter.

Som en sista del av litteraturstudien undersöks ämnet stilometri att för att förbereda inför programmeringen. Dels för att få en djupare förståelse för ämnet och dels för att

konstatera vilka egenskaper hos texten som kan analyseras. Hur dessa egenskaper analyseras rent programmeringsmässigt och huruvida de är praktiskt genomförbara behandlas inte under denna del. Denna del hålls teoretisk, medan de praktiska aspekterna hanteras under planeringsfasen av programmeringen.

Experiment I - Simulering

Den andra delen av projektet är en praktisk del med tester av detektorer för genererade texter. De detektorer som framkommit under kartläggningen, både från den egna kartläggningen och från tidigare arbeten, testas. Från "A Thesis that writes itself" [3] så testas två detektorer, GLTR [4] och RoBERTa [5] Resultaten av tidigare arbeten jämförs med aktuella resultat och båda ställs gentemot vad som är statistiskt sannolikt.

Resultaten presenteras dels med hjälp av diagram där skillnaderna i resultaten presenteras. Utgångspunkten var att genom hypotestest med metoden Chi Square se om de uppmätta skillnaderna är signifikanta, men baserat på resultaten så finns det inget behov av detta. Resultaten är i sig intressanta för att bedöma de olika verktygens träffsäkerhet i olika situationer, men de hjälper även till under framtagningen av den egna testmetoden.

För att testa detektorerna genereras text med den senaste publikt tillgängliga versionen av ChatGPT. Det går att tänka sig att det finns ett behov av att generera texter från olika personer, inte minst med tanke på de relativt nya funktionerna hos ChatGPT att den väger in gamla frågor och svar i nya frågor [6]. Det vill säga att man låtit flera personer från olika geografiska platser skriva texter för att minimera risken för att lokala språkliga egenheter kontaminerar statistiken. Dock visar de preliminära resultaten att det inte fanns något behov av detta.

Även om denna uppsatts är skrivet på svenska så är texterna på engelska, delvis för att matcha tidigare arbeten inom området och även för att kunna jämföra texter skrivna av personer i olika delar av världen.

Resultaten från testningen tas i beaktning under programmeringen för att välja ut vilka parametrar programmet ska analysera utifrån. Detta görs baserat på hur redan existerande detektorer arbetar men även hur de nämnda generatorerna för texter skapar sin utdata.

Det är under testningen som frågeställningens andra fråga kan besvaras, hur bra är de sätt som idag finns för att detektera genererade texter.

Experiment 2 - Prototyper

Den tredje delen av projektet är en praktiskt del med syfte att skapa två prototyper av metoder för att detektera genererade texter med hjälp av statistik och stilometri. Med hjälp av den sammantagna kunskapen från kartläggningen och testerna så skapas en testmodell med vars hjälp som de redan existerande detektorerna under experiment ett.

Den första prototypen för att detektera genererade texter analyserar texten baserat på analys av ordlängd, meninglängd, ordfrekvens, viktade ordval, mängden bisatser, ordklasser såsom possessiva pronomen.

Den andra prototypen använder sig av stilometri biblioteket Fast Stylometry [7] och undersöker möjligheterna till att applicera stilometri för att detektera genererade texter. Denna prototyp kommer inte att gå in på djupet med kodningen utan hålla det enkelt för att se om konceptet är relevant.

För att testa den framtagna modellen så implementeras den i ett egenskapat program. Programmet tar in en text och efter en analys presenteras både grafer över hur denna text står sig mot andra texter samt hur sannolikt det är att texten är genererad. Eftersom detta program är ett första steg i att testa ett koncept för detektering av genererade texter så är fokuset på själva analysen i första hand, presentationen av resultaten i andra hand och i tredje hand användarvänlighet.

Valet av programspråk har övervägts noga och en kombination av skribentens kunskaper har ställts mot för- och nackdelarna hos olika programmeringsspråk. Kraven som ställts på programmeringsspråket är att det ska vara enkelt att hantera, enkel syntax, översiktligt och utan för mycket behov av kringliggande kod till den huvudsakliga koden. Då datamängderna är relativt små så ställs kraven för effektivitet i prestandan lågt. Koden ska vara replikerbar både i skrivande stund och goda möjligheter att replikera i framtiden och inte kräva någon speciell hård eller mjukvara. Det ska även ha möjligheten, antingen direkt eller via programmeringsbibliotek, att visa grafer.

Sammantaget har bedömningen lett till att implementeringen av modellen ska göras i programmeringsspråket Python då det uppfyller samtliga kriterier. Detta förutsätter att ett programmeringsbibliotek såsom Matplotlib [8] används för att generera den grafiska plottningen av statistiken. Den senaste versionen av programmeringsspråket Python används, som i skrivande stund är version 3.12.2. Att använda den senaste versionen gör det skrivna programmet mer framtidssäkrat.

Det är i denna sista del av uppsatsen som även den sista frågan i frågeställningen ska besvaras, där resultaten från den egna detektorn analyseras gentemot dels de egna

resultaten under testningen men även mot andras resultat för att kunna avgöra hur användbart Stilometri skulle kunna vara för att detektera genererade texter.

2.2 Problematisering

Ett scenario som kan tänkas uppstå är om de detektorer för genererade texter som finns idag skulle vara fullgoda för ändamålet. Men även om så är utfallet så besvarades ändå samtliga frågeställningens frågor och det egenutvecklade programmet skulle då bara påvisa att de existerande detektorerna uppfyller ändamålet.

Att litteratursökningen skulle missa någon välanvänd detektor för genererade texter under testerna är en möjlighet. Antingen för att den är publikt okänd eller inte påträffas under litteraturstudiens kartläggning över detektorer. En djupgående litteraturstudie och kontakt med personer som kan tänka sig använda sig av detektorer på till exempel Högskolan i Halmstad kan minska denna risk.

Att begränsa experimenten till ChatGPT kan mycket väl göra att den egenutvecklade metod för att detektera genererade texter inte är användbar på andra generatorer. Olika sätt att generera texten kan mycket väl ha sina egna egenheter, och det måste hållas i åtanke när resultaten analyseras.

Mängden indata i testerna är begränsad, vilket gör det svårt att dra konkreta slutsatser från resultaten. Och det ska inte heller dras några slutsatser gällande att detektera texter skrivna med tidigare eller senare versioner än den testade versionen.

När det kommer till programmeringen så kan valet av programmeringsspråk visa sig vara ett otillräckligt språk för ändamålet, och att detta upptäcks först när kodningen påbörjats. Då detta är ett medvetet val så togs det i beaktning när koden skrevs, för att koden ska vara generell nog för att kunna implementeras relativt enkelt i ett annat programmeringsspråk om så skulle vara fallet.

Utöver detta så görs ett test redan inledningsvis för att skapa vad som kan kallas ett proof-of-concept eller en minimum-viable-product. Det vill säga en väldigt grundläggande kod som tar en text som indata och räknar antalet av varje ord i texten och presenterar detta som ett grafiskt stapeldiagram. En fungerande och lyckad kod går sedan att bygga vidare på och bygga upp de typer av analys som bedömstill ska göras för att detektera om en text är genererad.

Det finns även en reell möjlighet att resultaten från det egenutvecklade programmet inte påvisar att Stilometri är en användbar metod för att detektera genererade texter. Även om så är fallet så är även ett negativt utfall på experimentet intressant, då det kan hjälpa till att avfärda det som en användbar metod.

När det kommer till etiska aspekter så hanteras inga känsliga eller personliga uppgifter. Det rör inte heller någon form av kringgående av tekniska säkerhetssystem vilket hade haft en djupare etisk aspekt.

Det som kräver en etisk beaktning, är att skapa kod för att jämföra texter med stilometri. Och även om programmets huvudsyfte inte är att till exempel av-anonymisera en författare så skulle en modifierad version av koden kunna användas till att hjälpa till att göra detta, precis som det stilometri används till. Detta kommer att finnas i åtanke och kontrolleras innan det att källkoden publiceras.

Författaren av detta arbete har ingen koppling till tillverkarna av de generatorer och detektorer som testas. Författaren har inte heller blivit betald eller ombedd av någon att skriva detta arbete, utan gör det helt på eget initiativ. Författaren finner ingen vinning i om utfallen av resultaten av experimenten och programmeringen är positiva eller negativa. Sammantaget så finner författaren inga intressekonflikter gällande skrivandet av detta arbete.

3 Litteratursökning

Inledningsvis kartläggs ämnet för att fastställa hur och varför den genererar den text den gör, och konstatera om det finns någon intelligens i detta verktyg som ofta omnämns som AI, Artificiell Intelligens. En klar bild som utgångspunkt främjar nya synsätt på processen med att utveckla och testa metoder för detektering.

Utöver att kartlägga ämnet och öka förståelsen för den är målet med denna litteratursökning ett tvådelat. Den ena delen är att undersöka vilka metoder som finns tillgängliga idag för att detektera genererade texter. Den andra relaterade delen är att undersöka om det går att använda sig av statistik och stilometri för att detektera sådana och ställa denna metod i kontrast med de redan tillgängliga. Utöver dessa mål är förhoppningen att informera om hur en språkmodell såsom ChatGPT fungerar, och därmed även de styrkor och svagheter en sådan har, för att bidra till en ökning av den allmänna kunskapen inom ämnet.

När det kommer till ”A thesis that writes itself: On the threat of AI-generated essays within academia” [3] så ställs frågan om är hur situationen ser ut nu två år senare. Har genereringen blivit mer naturlig, har detektivernas träffsäkerhet ökat, eller har kapprustningen mellan generatorerna och detektorerna stagnerat?

Återfinns det en problematik hos de tjänster som idag säger sig kunna detektera genererade texter, och går det att ta fram en metod baserat på hur språkmodeller är uppbyggda som både är bättre än dessa och mänsklig granskning?

3.1 Tidigare arbeten

A thesis that writes itself

A thesis that writes itself [3] har ett fokus på hur bra människor, specifikt lärare och dylika personer, är på att detektera genererade texter och genererade texters påverkan på akademien.

En möjlig problematik med metodvalen är att de personer som testades på hur bra de var på att detektera genererade texter var medvetna om dels att de testades och på vad testet var. Om en person är medveten om att denne testas så finns det en möjlighet att de analyserar texten på ett annat sätt än om det var ett omedvetet test. Med den problematiken i åtanke så är ändå resultaten intressanta.

A thesis that writes itself [3] kom fram till att de som testades hade en näst intill 50% chans att gissa rätt huruvida en text var artificiellt genererad eller skriven av en människa. Vilket betyder att under ideala förutsättningar med någon som är van att bedöma texter, ändå bara är lika bra att bedöma om en text är genererad som en singlad slant.

Utöver detta så så undersökt det även på hur väl testpersonerna bedömer texter som inte är genererade. Där är gissningarna mer korrekta, men uppnår bara en träffsäkerhet på cirka 65%.

A thesis that writes itself [3] undersöker även detektorerna GLTR och en implementering av RoBERTa som detektor och hur dessa detekterar genererade texter. GLTR undersöker hur sannolikt det är att ett ord följs av ett annat med hjälp av GPT. Även om detta är en mycket intressant lösning, så har den flera svagheter. Dels kräver den tillgång till GPT, rimligtvis via en internetuppkoppling. Att låta generatören själv bestämma om en text är genererad skulle kunna skapa en konflikt, och uppdateringar av GPT bör förändra sannolikheten i ordföljerna. Till sist så är verktyget fokuserat just på GPT, som visserligen är vanligt förekommande, så bör den därför fungera desto sämre på texter som inte är skrivna med GPT. RoBERTa är en språkmodell som är tränad på GPT för att detektera hur lik en text är mot hur texter från GPT är. Problematiken med RoBERTa implementerade detektorn är i linje med den problematiken som också noteras hos GLTR.

En reflektion gällande båda dessa metoder att detektera genererade texter är att de är relativt tunga, och en enklare första sällning av arbeten hade fyllt ett behov. Dessutom att kunna göra denna första sällning i ett slutet system, det vill säga offline, hade även varit användbart speciellt om materialet har en nivå av upphovsrätt eller sekretess som gör det önskvärt att sprida det.

Detektering av fusk vid användning av AI

Denna uppsats, Detektering av fusk vid användning av AI: En studie av detektionsmetoder [9], fokuserar på att detektera genererade texter inom akademien och fuskande inom skolväsendet. Denna uppsats lyfter fram de begränsningar som finns med ChatGPT, där ibland att generatören kan ge "meningslösa svar" och att den överanvänder vissa termer.

Uppsatsen ser över olika typer av metoder för detektering, där den ena, "Säck av ord", kommer att användas i Experiment 2 - Prototyp 1. Metodens syfte uppfattas framförallt vara för att detektera plagiat. Detektering av fusk vid användning av AI nämner även Stilometri som en möjlig metod, vilket kommer att användas i Experiment 2 - Prototyp 2.

Förutom att undersöka text som "A Thesis that Writes itself" [3] gör, så undersöker Detektering av fusk vid användning av AI [9] även programmeringskod och matematik. Och även om detta inte är direkt relaterat till detta detektering av generade texter så kan de ändå ge en indikation på vilka aspekter av genereringen som skulle kunna detekteras.

Uppsatsen går även in på att ChatGPT har problem när det kommer till matematik och att komma till rätt svar, vilket stödjer det iakttagande som ligger i grund för detta arbete, frågan om varför ChatGPT räknar fel.

Uppsatsen har valt ut två detektorer för genererade texter, GPTZero [10] och AI Text Classifier [11].

Uppsatsen summerar resultaten som att programkod är enkelt för en person att urskilja om den är artificiellt genererad. Däremot så noteras att det är “särskilt utmanande” för människor att detektera matematik och text som är artificiellt genererad.

3.2 Språkmodeller

Hur genererar en språkmodell sina texter?

Texter från en språkmodell genereras genom en sannolikhetsfördelning av ordningen på orden. En stor språkmodell, Large Language Model, agerar som maskininlärning med en stor mängd text. Källan på dessa texter kan vara publicerade texter eller hemsidor [12].

Artificiell Intelligens (AI)

En egen reflektion som framkommit under litteraturstudien är att även om användningen av ord förändras så har utbredningen av ordet AI nått en nivå där väldigt mycket får den stämpeln. Problematiken som framkommer på grund av detta är relaterad till citaten som frågeställningen delvis baseras på, att den generella kunskapen om vad en språkmodell faktiskt klarar av är bristfällig. Att det finns en övertro på att språkmodeller är smarta och att de tänker och skapar nya egna saker.

Djupinlärning (Deep learning) är en undergrupp till Maskininlärning (Machine Learning) som i sin tur är en undergrupp till Artificiell Intelligens (Artificial Intelligence), vilket betyder att det tekniskt sett är korrekt att kalla en språkmodell för en AI, men eftersom språkmodeller endast är en sån liten del av vad en AI teoretiskt kan vara kapabel till att göra så är det enkelt att anta att någon som benämns som en AI har större kapabilitet än vad den faktiskt har [13].

Det huvudsakliga syftet med denna utläggning om användandet av orden Artificiell Intelligens är dock inte för att diskutera huruvida det är korrekt att använda eller huruvida det skapar en övertro hos verktygen, utan om kunskapen om att dessa verktyg inte är smarta utan genererar texter ord för ord baserat på de texter de är tränade på. Detta kan skapa en möjlighet att detektera genererade texter med till exempel Stilometri då texterna skulle kunna bli väldigt lika.

Fel i genererade texter

Eftersom en språkmodell inte tänker, inte är någon intelligens, utan en kombination av statistik och mönster så kontrollerar inte generatorm sin egen text och är begränsad till de källor den haft tillgång till under inlärningen [12]. Om källorna dessutom har olika information så behöver språkmodellen vikta dessa, och kombinationen av olika källor kan resultera i information som inte är sammanhängande eller talar för samma slutsats.

En språkmodell genererar texter, vilket betyder att när det kommer till matematiska uträkningar så räknar den inte, utan genererar en text som råkar innehålla matematik. Detta kan resultera i att matematiska uträkningar blir fel.

Detta betyder att språkmodellen är starkt baserad på dess träningsdata och inte dubbelkollar den genererade texten successivt i takt med att den skapas, den saknar

alltså en logik att följa i den genererade texten. Den är begränsad till att generera text utifrån givna förutsättningar.

Sammantaget så är dessa egenskaper hos en språkmodell just de som detta arbete hoppas kunna använda sig av för att detektera om en text är genererad eller skriven av en person.

Hur detekteras texter i dagsläget?

Om vi ser till GLTR och implementeringen av RoBERTa som undersöks i “A Thesis that Writes itself” [3] så kollar båda dessa detektorer på hur sannolikt det är att ett ord i en text efterföljs av ett annat om de vore genererade. De använder sig därmed av den kunskapen som vi tidigare fastställt, att generatorerna genererar ord för ord. Det som skiljer dessa verktyg åt är hur denna statistik tas fram, där GLTR använder GPT för att se vad den hade genererat för nästa ord och RoBERTa är en språkmodell som är tränade på texter från GPT.

När det kommer till detektorerna som används i “Detektering av fusk vid användning av AI” [9], så är dessa AI Text Classifier och GPTZero. AI Text Classifier är utvecklat av OpenAI och har tränats på genererade texter från en stor mängd generatorer, Wikipedia samt OpenAIs tidigare system. Det rekommenderas att använda och testa engelska texter. GPTZero är även den en detektor som tränats på en stor mängd texter, både genererade och skrivna. Den använder sig av två mätvärden, Perplexity och Burstiness. Perplexity avgör hur pass förutsägbar texten är och Burstiness undersöker likheter kontra variation i meningar. Det påpekas även att ju längre en text är desto bättre blir bedömningen.

Detta upplägg för att detektera genererade texter är intressant men kommer med sina begränsningar och svagheter. Dels uppdateras generatorer för texter hela tiden och vid tillräckligt stora uppdateringar så finns risken att gammal träningsdata inte längre är aktuell, eller att språkmodellen genererar andra svar på vilket kommande ord som är mest troligt. Då processen är relativt tung och långsam, så lämpar den sig sämre om man har stora mängder med text att undersöka.

De kräver dessutom en uppkoppling mot de aktuella språkmodellerna vilket då både har tekniska krav men även kan vara problematiskt om det kommer till känsliga eller upphovsrättsskyddade texter.

3.3 Stilometri

När det kommer till applicering av stilometri så är det en statistisk analys av lingvistik. Baserat på det som framkommit under denna litteratursökning så finns det potential att använda detta för att detektera genererade texter. Det finns alltså egenskaper hos text som detta arbete identifierat att vara aktuellt för att analysera potentiella genererade texter.

Ett användningsområde för stilometri är att fastställa en författare på en omstridd text, och sådan applicering för att jämföra en individs texter med andra texter denna skrivit

för att undersöka liknelser är något som kan vara intressant när det kommer till texter inom akademien.

När det kommer till de mer språkliga aspekterna så kommer mängden ord i ordklassen pronomen att analyseras. Fokuset kommer att vara på undergruppen Possessiva pronomen, men övriga typer av pronomen kommer att tas i beaktning beroende på om undersökningen av possessiva pronomen uppvisar någon potential. De andra undergrupperna av pronomen är Demonstrativa, Determinativa, Indefinita, Interrogativa, Personliga, Reciproka, Reflexiva och Relativa.

4 Experiment I - Simulering

4.1 Tester

Från “A thesis that writes itself” [3] testades detektorerna GLTR och RoBERTa.

Från “Detektering av fusk vid användning av AI: En studie av detektionsmetoder” [9] valdes endast GPTZero ut för tester. Detta baserat på att AI Text Classifier inte längre är aktiv.

För testerna analyserades korta texter på en paragraf genererade av ChatGPT, längre texter på tio paragrafer genererade av ChatGPT, egenskrivna texter med ett A4:s längd, och klassiska litterära verk.

5 Experiment 2 - Prototyper

Utifrån den information som framkommit under litteratursökningen så hade en enklare detektor för genererade fyllt ett potentiellt behov hos de som vill detektera genererade texter. Att dels snabbare kunna sålla bland texter för att sedan göra en djupare analys och även kunna detektera utan att behöva skicka iväg de aktuella texterna någonstans.

Syftet med dessa prototyper är dels att se vilka avvikande egenskaper det går att finna i detekterade texter men även om det går att använda stilometri för samma ändamål.

Valet av programmeringsspråk har gjorts till Python av senaste versionen baserat på överläggningarna i metoddelen. Utvecklingen av koden skrivs i miljön Visual Studio Code på en PC med Microsoft Windows 10.

Två prototyper tas fram för att undersöka om stilometri går att applicera för att detektera om flera arbeten har samma författare, det vill säga om de är skrivna av en generator eller av en människa. Den ena prototypen är av egen design för att se om det går att detektera texter utan att behöva bygga ett referensbibliotek och den andra använder ett färdigt bibliotek för att bygga ett enklare lokalt bibliotek. Tanken med den andra prototypen är om man har en fast uppsättning av något man vill kontrollera.

Den första prototypen är ett enklare egenutvecklat test där ordlängd, meningslängd, ordfrekvens, vikta ordval, mängden bisatser och ordklasser såsom possessiva pronomen undersöks för att se om det kan ge en indikation på huruvida en text är genererad eller ej. Det som hade varit en styrka med detta test är att det går att köra lokalt, är lättviktigt och det kräver ingen träningsdata för att ge ett preliminärt resultat.

Den andra prototypen kommer att använda sig av Python-biblioteket Faststylometry för att göra en djupare stilometrisk undersökning av texter. Denna typ av test går att köra lokalt och är lättviktigt, men det har ett något högre resurskrav på att kunna jämföra texter mot varandra.

5.1 Prototyp I

För att strukturera upp programmeringen så behöver behoven av programmets funktionalitet listas upp. Programmet behöver läsa in text, bearbeta den inmatade texten, göra jämförelser och presentera ett resultat. Sammantaget så presenteras kravspecifikationen nedan. Kraven är uppdelade i tre kategorier; behov, önskvärt och extra. Behov är det minsta som krävs för grundläggande funktionalitet i programmet. Under kategorin önskvärt är sådant som hade ökat funktionaliteten på programmet men inte krävs för dess grundläggande funktionsbehov. Extra är sådant som identifierats som användbart men inget krav och går att bortse ifrån om det inte faller inom den givna tidsramen. Programmeringen kommer att följa ett naturligt flöde, och funktioner som inte anses vara grundläggande kan komma att skapas före på grund av oavsiktliga eller oplanerade enkelt implementerade lösningar. Källkoden återges i sin helhet i Bilaga A.

Om vi först ser till de rent tekniska aspekterna av text när det kommer till Stilometri så kommer meningslängden att analyseras, dels antalet ord per mening men även antalet bokstäver per mening. När det kommer till ordlängd så är både snittet på antalet bokstäver per ord intressant, men även snittet på antalet bokstäver per ord per mening. Den sista mer tekniska aspekten är de egenheter som text uppvisar i form av antalet bisatser, upprepningar av ord och antalet unika ord.

Sammanfattningsvis är de intressanta parametrar som framkommit att testa är antalet meningar, antalet ord, antalet tecken, antalet bi-satser, unika ord, procentuellt sett hur mycket unika ord, genomsnittligt antal ord per mening och genomsnittligt antal bokstäver per ord.

Kravspecifikation

Behov

- Ta emot indata via en variabel.
- Bearbetning av texten enligt de parametrar som framkommit under kartläggningen och experimenten.
- Presentera resultat.

Önskvärd

- Ta emot indata från en fil.
- Presentera grafer över resultat.

Extra

- Ta emot indata från samtliga textfiler i en given mapp.
- Fler parametrar som framkommer under programmeringen.

5.2 Prototyp 2

Denna prototyp följer samma upplägg som den första prototypen men kommer att använda sig av kodbiblioteket Faststylometry för Python. På grund av kompatibiliteten på Faststylometry så har version 3.10 av Python använts för denna delen av koden. Baserat på detta biblioteks möjligheter och färdiga exempelkod så har kravspecifikationen justerats och förenklats gentemot den första prototypen. Källkoden återges i sin helhet i Bilaga B.

Kravspecifikation

Behov

- Bearbetning av texten.
- Presentera resultat.
- Ta emot indata från samtliga textfiler i en given mapp.

6 Resultat

6.1 Test av detektorer

Test av GLTR

Denna metod för detekterings standardinställning är att klassificera ord i tre kategorier beroende på hur stor chansen är att GPT skulle generera det som nästa bokstav.

Färgkodningen innebär en ökning av magnitud på antalet genereringar det ordet skulle förekomma. Där grön är 1 på 10, gul är 1 på 100, röd är 1 på 1000 och lila färre än så.

Utifrån de demonstrationsexempel som GLTR tillhandahåller har GPT-2 genererade texter nästan uteslutande grön med inslag av gult. En mänskligt skriven text har fler inslag av röda och lila. När en nyskriven text från ChatGPT matas in så ges ett resultat mycket likt de resultat som återfinns när mänskliga texter matas in.



Space, the final frontier, represents the vast and uncharted expanse beyond Earth's atmosphere, encompassing the celestial bodies, galaxies, and cosmic phenomena that populate the universe. From the mesmerizing beauty of distant nebulae to the enigmatic allure of black holes, space captivates the human imagination with its boundless mysteries and infinite possibilities. Exploring space not only expands our scientific knowledge but also deepens our understanding of the cosmos and our place within it.

The study of space, known as astronomy, traces its roots back to ancient civilizations that gazed up at the night sky in awe and wonder. Over millennia, astronomers have observed, cataloged, and theorized about the stars, planets, and galaxies that populate the universe, unraveling the secrets of cosmic evolution, stellar formation, and the origins of the universe itself.

Space exploration represents humanity's quest to explore, discover, and unlock the mysteries of the cosmos. From the first tentative steps into space during the Space Age of the 20th century to ambitious missions to the moon, Mars, and beyond, space exploration pushes the boundaries of human achievement and technological innovation. Space agencies such as NASA, ESA, and Roscosmos lead the charge, launching spacecraft, rovers, and telescopes into the depths of space to explore distant worlds and unravel the mysteries of the universe.

The exploration of space has yielded groundbreaking discoveries that have reshaped our understanding of the cosmos. From the discovery of exoplanets orbiting distant stars to the detection of gravitational waves rippling through space-time, each new revelation expands our knowledge and challenges our assumptions about the nature of the universe. Space serves as a natural laboratory for scientific inquiry, enabling researchers to conduct experiments and observations in microgravity environments that are impossible to replicate on Earth. Space-based research spans a wide range of disciplines, including astronomy, physics, biology, and medicine, offering insights into everything from stellar evolution and cosmic radiation to human health and aging.

Space exploration has practical applications that benefit humanity in myriad ways. Satellite technology enables global communication, navigation, weather forecasting, and disaster monitoring, while space-based observatories provide valuable data for scientific research and environmental monitoring. Space exploration also drives innovation in materials science, robotics, and propulsion technology, leading to technological advances that improve our daily lives and spur economic growth.

The colonization of space represents a bold vision for the future of humanity, offering the potential to establish permanent settlements on other planets and moons within our solar system and beyond. While still in its infancy, projects such as SpaceX's Starship and NASA's Artemis program aim to return humans to the moon and lay the groundwork for crewed missions to Mars and beyond. The colonization of space poses numerous challenges, from technological and logistical hurdles to ethical and philosophical questions about the nature of human exploration and settlement.

Space tourism represents a burgeoning industry that offers private citizens the opportunity to experience the wonders of space firsthand. Companies like SpaceX, Blue Origin, and Virgin Galactic are developing spacecraft and launch systems to transport paying customers on suborbital joyrides and orbital adventures, opening up space travel to a new generation of explorers and adventurers.

The search for extraterrestrial life represents one of the most compelling quests in space exploration. Scientists are actively searching for signs of life on other planets and moons within our solar system, such as Mars, Europa, and Enceladus, as well as distant exoplanets orbiting other stars. The discovery of microbial life on Mars or evidence of habitable conditions on other worlds would have profound implications for our understanding of the origins and prevalence of life in the universe.

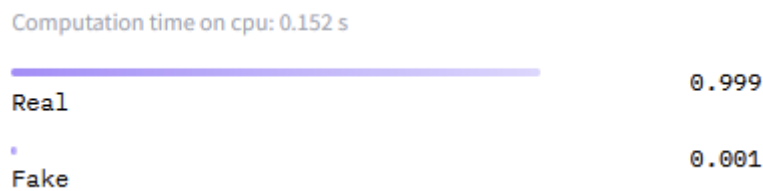
In conclusion, space represents humanity's greatest frontier, beckoning us to explore, discover, and dream of what lies beyond. Whether through scientific inquiry, technological innovation, or human exploration, the exploration of space offers endless opportunities to expand our knowledge, inspire future generations, and shape the destiny of our species among the stars.

Figur 1. Utdata från GLTR på 10 paragrafers text skiven av ChatGPT där grön är väntat, gult är mer oväntat, rött är väldigt oväntat och lila är mycket oväntat.

Test av RoBERTa

Problematiken med nya generationer av textgeneratorer är även framträdande vid testerna av RoBERTa. På de slumpmässigt utvald egenskriven texterna på ca 1 sida styck så ger den en sannolikhet på 99.9% att den inte är genererad. En text på ett stycke

från ChatGPT får en sannolikhet på 99.8%. RoBERTa hade en begränsnings på inmatad text, så de första sex paragraferna från en av de längre generade texterna från ChatGPT matades in och den gav en träffsäkerhet på 99.6%.

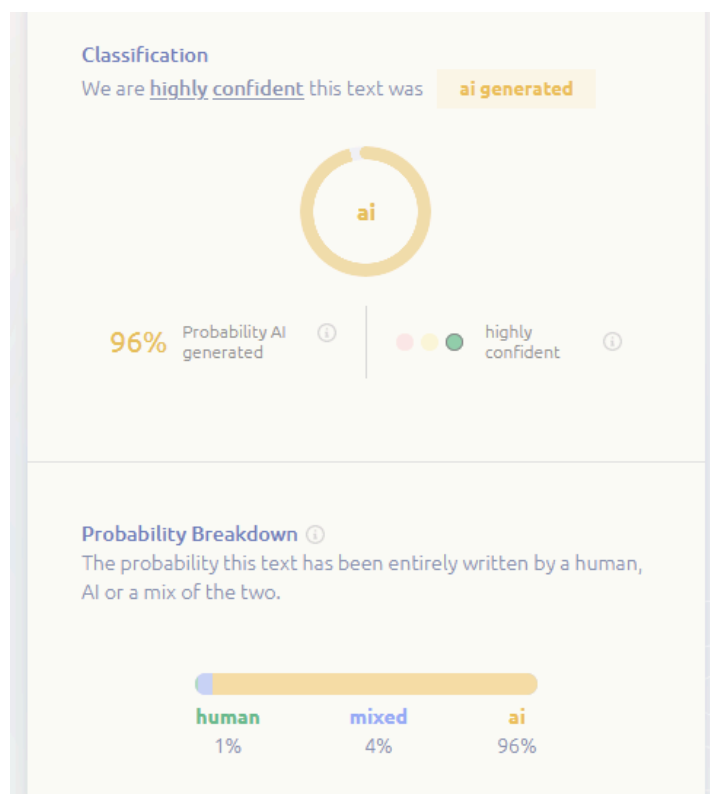


Figur 2. Utdata från RoBERTa på ca 1 A4 lång text skriven av författaren

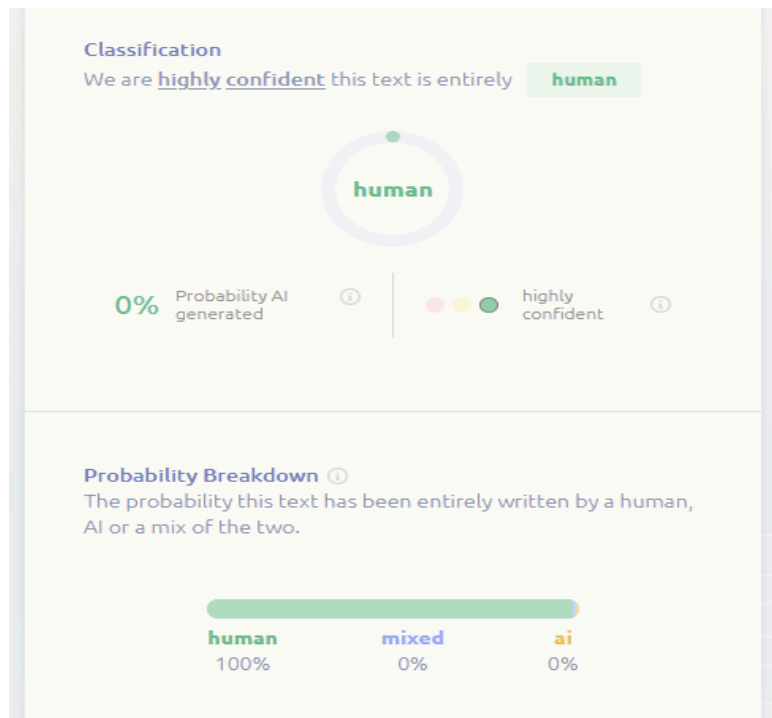
Test av GPTZero

De exempeltexter som GPTZero har ger en god träffprocent och När det kommer till korta texter genererade av ChatGPT på en paragraf så ger GPTZero dom ett span på 79%-98%.

När det kommer till de sex stycken egenskrivna texterna på ca en sida var så ger GPTZero en sannolikhet på 0% att dessa inte är skriva av en människa.



Figur 3. Utdata från GPTZero på 10 paragrafers text skriven av ChatGPT, där resultatet är en procent chans att en text är genererad.



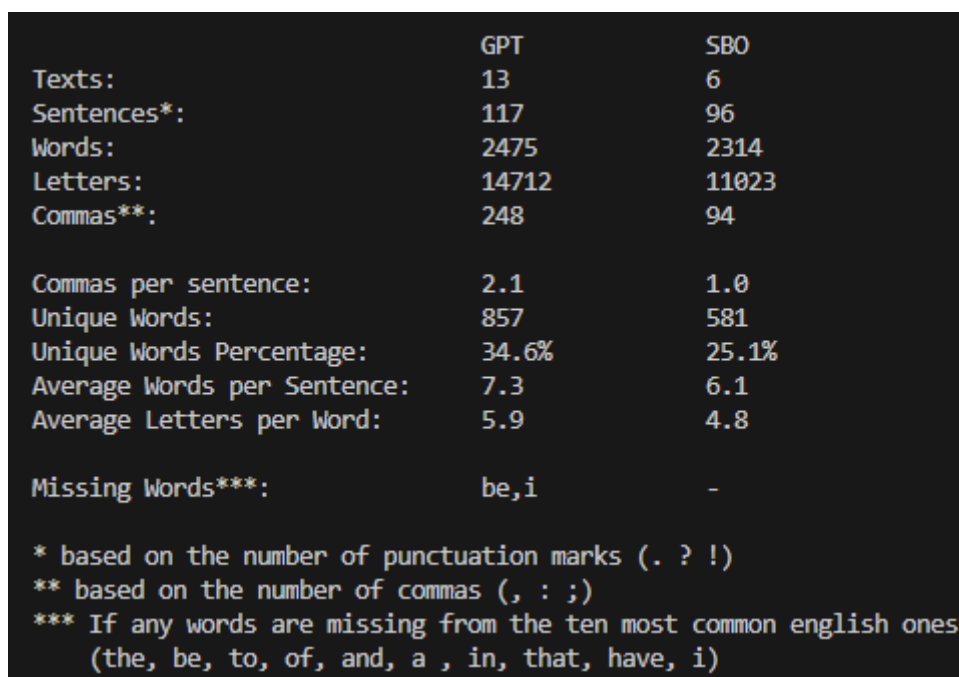
Figur 4. Utdata från GPTZero på 10 paragrafers text skriven av författaren, där resultatet är en procent chans att en text är genererad tillsammans med en säkerhet och nedbrytning av procent chansernas distribution.

6.2 Prototyp 1

En prototyp färdigställdes i programmeringsspråket Python som uppfyller samtliga Behov, Önskvärda och Extra egenskaper utefter kravspecifikationen.

Denna prototyp analyserar samtliga textfiler i en mapp och sorterar upp dem beroende på första ordet i filnamnet. Där GPT är texter genererade av ChatGPT och SBO är texter skrivna av författaren. Informationen i de inlästa filerna mäts och delas upp i flertalet kategorier som går att se i Figur 5. Valet av formatet på filnamnen baserades på det format som Faststylometri biblioteket använder som används i Prototyp 2.

Under programmeringen så framkom det en till parameter att analysera, det var de tio mest vanligt förekommande engelska orden i skriven text [10]. Detta går mer in i detalj under diskussionen.



```

GPT          SBO
Texts:       13          6
Sentences*:  117         96
Words:       2475        2314
Letters:     14712       11023
Commas**:    248         94

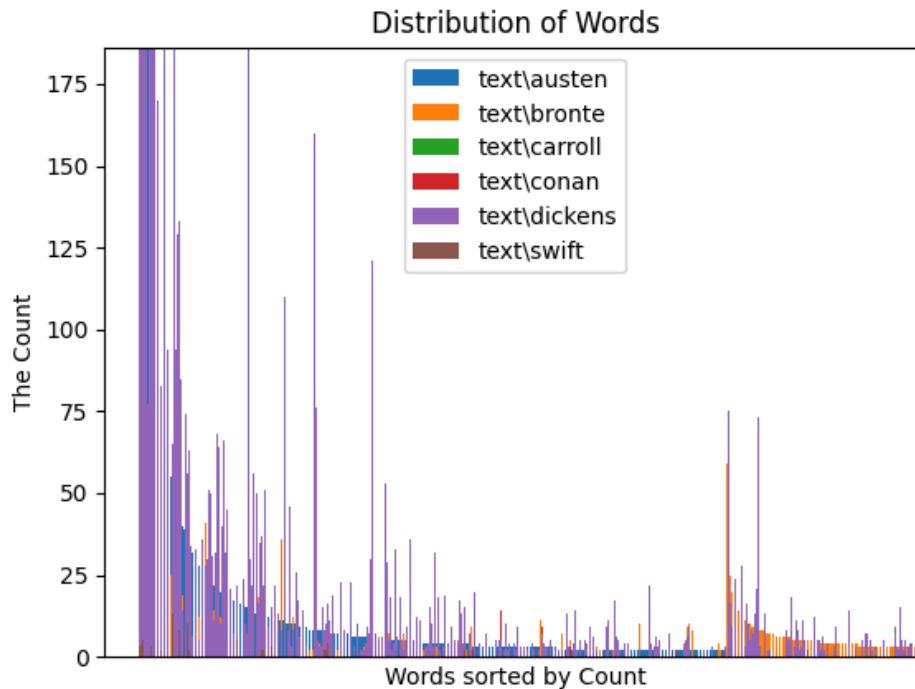
Commas per sentence:  2.1         1.0
Unique Words:         857         581
Unique Words Percentage: 34.6%       25.1%
Average Words per Sentence: 7.3         6.1
Average Letters per Word:  5.9         4.8

Missing Words***:     be,i         -

* based on the number of punctuation marks (. ? !)
** based on the number of commas (, : ;)
*** If any words are missing from the ten most common english ones
(the, be, to, of, and, a , in, that, have, i)
```

Figur 5. Exempel på utdata från Prototyp 1, där värden och beräknade värden presenteras tillsammans med eventuella ord som prototypen hittat som saknas. Kolumnen GPT är texter skriva av ChatGPT och kolumnen SBO är texter skriva av författaren. Den viktigaste skillnaden att notera är de ord som ChatGPT inte har med.

Från ChatGPT genererades det tretton texter. Tio av dessa genererades genom prompten “Please write me a paragraph about a flower. Den genererade en paragraf för blommorna cherryblossom, daffodil, daisy, hydrangea, lily, orchid, peony, rose, sunflower och tulip. Utöver detta användes prompten “Please write me ten paragraphs about X”, där X var flowers, minerals och space.



Figur 6. Exempel på grafisk utdata från Prototyp 1 där ordfrekvensen hos de olika författarna presenteras.

I Figur 6 återfinns ett exempel på den grafiska representationen av ord, bortsett från de tio vanligaste i engelsk skriven text och bortsett från unika ord. Orsaken till att dessa filtreras bort är att dessa antingen redovisas på andra ställen eller är av en så pass stor mängd att det gör det svårare att analysera övriga data. Baserat på detta utdata så fanns potentialen att några slutsatser skulle kunna dras, men som i exemplet så gör stora mängder text det svåröverskådligt.

Texten från fyra böcker av Jane Austen, två av Charlotte Brontë, en av Lewis Carroll, en av Arthur Conan Doyle, fem av Charles Dickens och en av Jonathan Swift användes.

	AUSTEN	BRONTE	CARROLL	CONAN	DICKENS	SWIFT
Texts:	4	2	1	1	5	1
Sentences*:	26217	27321	1883	7518	71947	295
Words:	451786	407789	29647	107603	1094565	6533
Letters:	2102562	1886400	138534	482418	4976685	32520
Commas**:	41846	41953	3018	8080	113055	575
Commas per sentence:	1.6	1.5	1.6	1.1	1.6	1.9
Unique Words:	17138	28114	3456	8575	35573	1355
Unique Words Percentage:	3.8%	6.9%	11.7%	8.0%	3.2%	20.7%
Average Words per sentence:	17.2	14.9	15.7	14.3	15.2	22.1
Average Letters per Word:	4.7	4.6	4.7	4.5	4.5	5.0
Missing Words***:	-	-	-	-	-	-

* based on the number of punctuation marks (. ? !)
 ** based on the number of commas (, : ;)
 *** If any words are missing from the ten most common english ones (the, be, to, of, and, a , in, that, have, i)

Figur 7. Utdata från Prototyp 1 där prototypen körs mot klassiska verk, där värden och beräknade värden presenteras tillsammans med eventuella ord som prototypen hittat

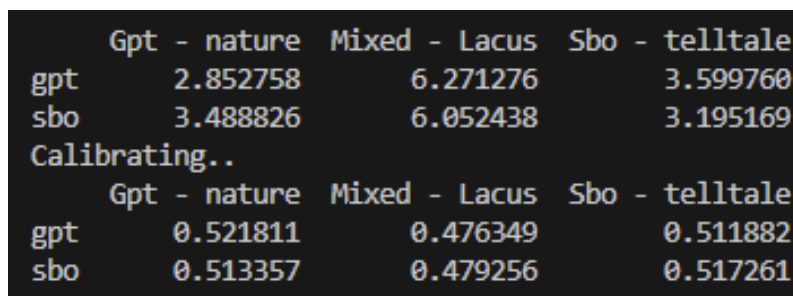
som saknas. Här noteras att inga ord saknas och att vissa värden varierar relativt kraftigt.

6.3 Prototyp 2

En prototyp färdigställdes i programmeringsspråket Python som uppfyller samtliga kravspecifikationens Behov. Koden baserades helt på den exempelkod som medföljer det aktuella kod-biblioteket [7].

Prototypen analyserar det givna indata från en mapp och presenterar resultat först i form av en sannolikhets gradering och sen en procentvärde efter en kalibrering mot indata.

I överkant av utdata så listas de testade filerna mot vilken författare den kopplar texterna till i vänsterkant. Filnamnen i överkant är döpta till om dessa kommer från ChatGPT, är en blandning eller är skrivna av författaren till detta arbete, detta är bara för att öka tydligheten på exemplen på utdata som presenteras.



	Gpt - nature	Mixed - Lacus	Sbo - telltale
gpt	2.852758	6.271276	3.599760
sbo	3.488826	6.052438	3.195169
Calibrating..			
	Gpt - nature	Mixed - Lacus	Sbo - telltale
gpt	0.521811	0.476349	0.511882
sbo	0.513357	0.479256	0.517261

Figur 8. Exempel på utdata från Prototyp 2, där första värdet anger hur avvikande texten är, där låga värden är lika, medan den andra är en procentchans där chansen viktats mot texternas innehåll. Här noteras att skillnaden mellan procentchansen hos ChatGPT och hos författaren är försumbar.

Denna prototyp med en applicerad Stilometri kördes mot tre typer av texter, korta, halvlånga och långa. Där korta var en paragraf, halvlånga tio paragrafer eller en sida, och långa skönlitterära böcker.

Uptill detta användes sex stycken egenskrivna texter på ca en sida styck, alla skrivna av författaren till detta arbete. Ytterligare en text genererad av ChatGPT skapades där grunden är skriven av en människa, för att sedan bett ChatGPT förbättra språket på denna, för att sedan korrigeras av en människa.

När det kommer till korta och halvlånga texter så gav programmet resultat runt 50%. Figur 8 testar 10 paragrafer från chatGPT med en sida skriven av författaren till detta arbete. Procent chanserna skiljer sig bara marginellt.

Tydliga resultat framkom bara på de långa texterna, där hela litterära verk jämförs med varandra och så gav stilometri tydliga resultat.

7 Diskussion

Sammantaget visar resultaten från experimenten att nivån på genererade texter är på en sådan nivå att det både är svårt och problematiskt att detektera. Delar av varje experiment gav intressanta insikter som är värda både att framhäva men även genererar intressant data att jobba vidare på i framtida arbeten.

Resultaten från den första prototypen gav inga tillräckligt distinkta resultat från de statistiska testerna för att dra några slutsatser annat än att de påvisar problematiken med detektering. En aspekt som framkom under testandet av prototypen var att det uppfattades som att ChatGPT undviker att använda vissa ord även på relativt många texter. Denna egenhet noterades först då en bugg i programmet upptäcktes under produktionen. Att de tio i text mest vanligt förekommande orden inte skulle förekomma var inget som togs i höjd för initialt.

Möjligheterna att använda stilometri för att detektera genererade texter visade sig mycket begränsad och påvisar problematiken med detektering. När det kommer till stora mängder text, såsom flertalet böcker, så är stilometri användbart, men vid mindre mängder text än det så visar sig stilometri inte applicerbart. Det finns ett klart användningsbehov av en detektor som utan träning skulle kunna sålla bland stora mängder text, även kortare texter.

Om resultaten som framkommit jämförs med de från "A thesis that writes itself" [3] så framkommer det hur utvecklingen av generatorer för text, såsom ChatGPT, skapar problem med detektering. Att program tränade för två år sen inte längre är användbara på texter genererade idag. Om vi ser framåt så kan ytterligare problem uppstå när det behövs träning på många olika versioner av generatorer, att samma text behöver kontrolleras av ett 10-tal versionen av bara ChatGPT gör processen tung, och skapar ytterligare problematik där olika versioner kan ge olika resultat. Hur ska dessa i sådana fall vägas mot varandra?

Att tränade detektorer av genererade texter är en färskvara skapar dessutom ett helt nytt problemområde med False Positives, det vill säga texter som felaktigt bedöms vara artificiellt genererade. Vi ser som exempel under experimenten att texter genererade utifrån GPT3 rapporteras som mänskligt skrivna av generatorer tränade på GPT2.

En parallell som kan dras är till automatiskt bildigenkänning, där en övertro på teknologin, och ett aktivt undvikande att inte följa rekommendationen att inte låta systemet göra automatiska beslut, skapar stora problem där långa processer dras igång med väldigt lite kontroll [15].

Redan för två år sen så var det problematiskt för människor att detektera texter, så behovet av metoder för att detektera genererade texter existerar. Slutligen så behövs resultaten av detektorn GPTZero lyftas fram, den påvisade en nästan osannolikt hög träffsäkerhet.

Problematiken som framträder när detektionen är svår är att mycket text kan skapas på kort tid, och därmed ge ett sken för att något nyligen påhittat är mer utbrett än vad det

faktiskt är. Och en generator såsom ChatGPT kan även ge missvisande svar, vilket dels är problematiskt när det kommer till att sprida felaktig information och dessutom kan det ge en person ökad bekräftelse bias [16].

Sen kan man ställa sig frågan om man kan lita på dessa detektorer, utan att veta i detalj på hur de är designade och tränade så är det inte otänkbart att det finns buggar eller bakdörrar som förändrar resultatet.

Ett annat möjligt problem som i teorin kan uppstå framöver är ett stagnerande av eget kreativt tänkande både när det kommer till skrivande och när det kommer till matematik. I förlängningen så skulle detta kunna resultera i en minskad kunskap inom områdena, vilket sänker möjligheten hos individen att avgöra om en text eller uträkning är rimlig.

7.1 Framtida arbeten

Det finns flertalet möjligheter att bygga vidare på upplägget i detta arbete. Den del som framträder mest som intressant att forska vidare på är hur ChatGPT medvetet verkar undvika vissa ord i sina svar. Det kan mycket väl vara så att den är så pass strikt i sin egen interna logik att den helt bortser från ord som i en text skriven av en människa mycket väl kan ha framkommit.

En aspekt detta arbete inte går in på, är att göra en studie idag hur väl människor är på att detektera generade texter vore ett intressant ämne att utforska och ställa i kontrast till de resultat som man framkom till 2022 i "A thesis that writes itself".

Det har bara förekommit en begränsad mängd indata med fokuset på att skapa och testa ett koncept för detektering. Resultaten från liknande tester med stora mängder indata vore intressanta att analysera. Det är inte otänkbart att storskaliga tester skulle kunna komma andra resultat eller komma till nya insikter.

Även om det system för att generera text i detta arbete, ChatGPT, är vanligt förekommande så finns det flertalet andra som har andra datakällor och metoder att generera text som kan förändra möjligheterna att detektera med de metoder som används med ChatGPT.

Andra egenskaper, som kan vara svåra att implementera, men ändå är värda att ta i beaktning, kom på tal under en diskussion med universitetslektor Erik Järpe. Dels om det är möjligt att hitta en metod för att detektera hur mycket texten håller sig till ämnet, hur pass den håller sig kvar vid den så kallade röda tråden. Men även hur neutral texten förhåller sig till ämnet, samt om det är möjligt att hitta andra egenheter i texterna som går att detektera.

Ytterligare en vinkel som är värd att utforska är att göra liknande tester med fler detektorer av genererade texter, analysera dessa och dra slutsatser utefter vad dessa ger.

8 Slutsatser

Sammanfattningsvis så finns det idag metoder som utger sig för att detektera genererade texter men tillförlitligheten på dessa är i flera fall bristfällig. En bidragande faktor i den bristfällande tillförlitligheten kommer sig av att generatorerna, framförallt ChatGPT är i ständig utveckling med stora skillnader mellan texter genererade med GPT2 och GPT3, och att många av dagens detektorer kräver att de tränas på stora mängder indata.

En detektor som gav resultat över förväntan var GPTZero. De bedömer själva att “GPTZero is the leading AI detector for checking whether a document was written by a large language model such as ChatGPT.”[17].

Tidigare arbeten har påvisat att människor har svårt att särskilja genererade texter även under ideala förhållanden med förra versionen av GPT, vilket resulterar i ett behov av detektorer.

Att detektera genererade texter med hjälp av stilometri visade ha sina begränsningar, så pass att det i normalfallet inte är applicerbart. De genererade texterna är för lika texter skrivna direkt av människor för att hitta en märkbar skillnad.

Ett visst hopp kommer med den upptäckt som gjordes med den första prototypen: generatören ChatGPT undviker vissa ord i texter.

Referenser

- [1] Europaparlamentet (2020, 2023). “Artificiell intelligens: Möjligheter och risker”. <https://www.europarl.europa.eu/topics/sv/article/20200918STO87404/artificiell-intelligens-mojligheter-och-risker> [2024-03-25].
- [2] OpenAI (2022). “Introducing ChatGPT”. <https://openai.com/blog/chatgpt> [2024-03-25].
- [3] A. Olsson och O. Engelbrektson (2022). “A thesis that writes itself”. Kandidatuppsats. Högskolan i Halmstad.
- [4] H. Strobelt och S. Gehrmann (2024). “Catching a Unicorn with GLTR: A tool to detect automatically generated text”. <http://gltr.io/> [2024-03-22].
- [5] OpenAI (2024). “RoBERTa Detector”. <https://huggingface.co/openai-community/roberta-base-openai-detector> [2024-03-22].
- [6] OpenAI (2024). “Memory and new controls for ChatGPT”. <https://openai.com/blog/memory-and-new-controls-for-chatgpt> [2024-03-22].
- [7] Thomas Wood (2023). “Fast Stylometry Python Library”. <https://fastdatascience.com/fast-stylometry-python-library/> [2024-03-24]
- [8] The Matplotlib development team (2023). “Matplotlib”. <https://matplotlib.org/> [2024-03-24]
- [9] K. Ennajib och T. Liang (2023). “Detektering av fusk vid användning av AI: En studie av detektionsmetoder”. Fristående kandidatuppsats. Kungliga Tekniska Högskolan.
- [10] E. Tian, A. Cui (2023). “GPTZero: Towards detection of AI-generated text using zero-shot and supervised methods”. <https://gptzero.me> [2024-03-22].
- [11] OpenAI (2024). “Open AI Platform”. <https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text> [2024-03-22].

[12] AIContentfy team (2023). “How ChatGPT is Changing the Game for Text Generation“.
<https://aicontentfy.com/en/blog/how-chatgpt-is-changing-game-for-text-generation-1>

[13] IBM Data and AI Team (2023). “AI versus machine learning versus deep learning versus neural networks: What’s the difference?”.
<https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks>

[14] Wiki (2024). “Most common words in English”.
https://en.wikipedia.org/wiki/Most_common_words_in_English [2024-03-22]

[15] Sudhin Thanawala (2023) “ Facial recognition technology jailed a man for days. His lawsuit joins others from Black plaintiffs”.
<https://apnews.com/article/mistaken-arrests-facial-recognition-technology-lawsuits-b613161c56472459df683f54320d08a7>

[16] B. Casad, J.E. Luebering (2024). “Confirmation Bias”.
<https://www.britannica.com/science/confirmation-bias> [2024-04-02]

[17] GPTZero (2024). “GPTZero | Frequently asked questions”. <https://gptzero.me/faq> [2024-03-24]

Bilaga A - Källkoden Prototyp I

```
## Script: Cuppsatts.py
## Author: SB0
## Created: 2024-02-25
## Modified: 2024-02-25
## Purpose: Stilometry

import matplotlib.pyplot as plt
from collections import Counter
import os
import glob

os.system('cls')
print("\n")
textFiles = list(glob.glob("text/*.txt"))

authors = []
mostCommonEnglishWords =
["the", "be", "to", "of", "and", "a", "in", "that", "have", "i"]

statZero = "\t\t\t\t"
statFirst = "Texts:\t\t\t\t"
statSecond = "Sentences*:\t\t\t\t"
statThird = "Words:\t\t\t\t"
statFourth = "Letters:\t\t\t\t"
statFifth = "Commas**:\t\t\t\t"
statSix = "Commas per sentence:\t\t\t\t"
statSeven = "Unique Words:\t\t\t\t"
statEight = "Unique Words Percentage:\t\t\t\t"
statNine = "Average Words per sentence:\t\t\t\t"
statTen = "Average Letters per Word:\t\t\t\t"
statEleven = "Missing Words***:\t\t\t\t"
statDetailOne = "* based on the number of punctuation marks (. ? !)"
statDetailTwo = "** based on the number of commas (, : ;)"
statDetailThree = "*** If any words are missing from the ten most common
english ones"
```

```

statDetailFour = " (the, be, to, of, and, a , in, that, have, i)"

wordCount = {}
missingCommonWords = []
numberOfTexts, numberOfSentences, numberOfWords, numberOfUniqueWords,
numberOfCommas, numberOfLetters = 0,0,0,0,0,0

for textFile in list(textFiles):
    firstPart = textFile.split("_")[0]
    if firstPart not in authors:
        authors.append(firstPart)

for author in authors:
    wordCount.clear()
    missingCommonWords.clear()
    numberOfTexts, numberOfSentences, numberOfWords, numberOfUniqueWords,
    numberOfCommas, numberOfLetters = 0,0,0,0,0,0

    for textFile in list(textFiles):
        if textFile.split("_")[0]==author:
            fileToOpen = open(textFile, 'r', errors='ignore')
            textFileLines = fileToOpen.readlines()
            numberOfTexts += 1
            for textFileLine in textFileLines:
                numberOfSentences += textFileLine.count('.')
                numberOfSentences += textFileLine.count('?')
                numberOfSentences += textFileLine.count('!')
                numberOfCommas += textFileLine.count(',')
                numberOfCommas += textFileLine.count(':')
                numberOfCommas += textFileLine.count(';')
                singleWords = textFileLine.split()
                for singleWord in singleWords:
                    if not singleWord.lower() in wordCount.keys():
                        wordCount[singleWord.lower()]=1
                    else:

wordCount[singleWord.lower()]=wordCount[singleWord.lower()+1

```

```

        numberOfWords += 1
        numberOfLetters += len(singleWord)

for commonWord in mostCommonEnglishWords:
    if not commonWord.lower() in wordCount.keys():
        missingCommonWords.append(commonWord)

missingWords = ""
if len(missingCommonWords)>0:
    missingWords = ','.join(missingCommonWords)
else:
    missingWords = "-"

numberOfUniqueWords += Counter(wordCount.values())[1]

##Cache the output
statZero += str(author[5:]).upper() + "\t\t"
statFirst += str(numberOfTexts) + "\t\t"
statSecond += str(numberOfSentences) + "\t\t"
statThird += "\t" + str(numberOfWords) + "\t"
statFourth += str(numberOfLetters) + "\t\t"
statFifth += str(numberOfCommas) + "\t\t"
statSix += str(round((numberOfCommas/numberOfSentences),1)) + "\t\t"
statSeven += str(numberOfUniqueWords) + "\t\t"
statEight += str(round((numberOfUniqueWords/numberOfWords)*100,1)) +
"% " + "\t\t"
statNine += str(round((numberOfWords/numberOfSentences),1)) + "\t\t"
statTen += str(round((numberOfLetters/numberOfWords),1)) + "\t\t"
statEleven += str((missingWords)) + "\t\t"

##Prepare the graphics
##Remove 10 most common english words
wordCountfiltered = wordCount
for commonWord in mostCommonEnglishWords:
    if commonWord in wordCountfiltered.keys():
        del wordCountfiltered[commonWord]

```

```

    ##Remove unique words
    wordsToDelete = []
    for wordLeft in wordCountfiltered.items():
        if(wordLeft[1]==1):
            wordsToDelete.append(wordLeft[0])

    for deleteThis in wordsToDelete:
        if deleteThis in wordCountfiltered.keys():
            del wordCountfiltered[deleteThis]

    wordCountsorted = dict(sorted(wordCountfiltered.items(), key=lambda x:
x[1],reverse=True))

    plt.bar(list(wordCountsorted.keys()),list(wordCountsorted.values()))
    ##END Prepare the Graphics

##Printe the statistics
print(statZero)
print(statFirst)
print(statSecond)
print(statThird)
print(statFourth)
print(statFifth + "\n")
print(statSix)
print(statSeven)
print(statEight)
print(statNine)
print(statTen + "\n")
print(statEleven + "\n")
print(statDetailOne)
print(statDetailTwo)
print(statDetailThree)
print(statDetailFour)

print("\n")

```

```
##Show the graphics
plt.legend(authors)
plt.xlabel('Words sorted by Count')
plt.xticks([])
plt.ylabel('The Count')
plt.title('Distribution of Words')
plt.show()
```


Bilaga B - Källkoden Prototyp 2

```
## Script: CuppsattsLibrary.py
## Author: SBO
## Created: 2024-02-25
## Modified: 2024-02-25
## Purpose: StilometryLibrary

import os

os.system('cls')

from faststylometry import Corpus
from faststylometry import load_corpus_from_folder
from faststylometry import tokenise_remove_pronouns_en
from faststylometry import calculate_burrows_delta
from faststylometry import predict_proba, calibrate
import nltk

train_corpus = load_corpus_from_folder("trainGPT")
train_corpus.tokenise(tokenise_remove_pronouns_en)
test_corpus = load_corpus_from_folder("testGPT", pattern=None)
test_corpus.tokenise(tokenise_remove_pronouns_en)
print(calculate_burrows_delta(train_corpus, test_corpus, vocab_size = 50))
print("Calibrating..")
calibrate(train_corpus)
print(predict_proba(train_corpus, test_corpus))
```