Bruk av generative språkmodeller til å trekke ut geografisk informasjon om hendelser

Forfatter: Roald André Kvarv

Veileder: Bjørnar Tessem

Masteroppgave
Informasjonsvitenskap
Universitetet i Bergen

2. juni 2024

Sammendrag

Denne avhandlingen fokuserer på utviklingen av programvare for å ekstrahere geospatiale data fra nyhetsartikler ved hjelp av GPT-3/4 og lagre det i en kunnskapsgraf. Målet er å løse utfordringen med å trekke ut presis informasjon om steder, og geografiske forhold fra ustrukturerte tekster. Ved å utnytte avanserte tekstanalyse- og kunstig intelligensmetoder, ønsker prosjektet å støtte journalister og bidra til informasjonsanalyse.

Avhandlingen er godt tilpasset min bakgrunn i Informasjonsvitenskap og min interesse for tekstanalyse og maskinlæring. Det ligger et stort potensial for innovasjon innenfor automasjons-journalistikk. Mitt bidrag vil være å utvikle programvare som kan ekstrahere geospatiale data og etablere relasjoner mellom denne informasjonen i en kunnskapsgraf.

Jeg har undersøkt ulike geospatiale ontologier for å identifisere sammenhenger og mønstre lagrede data. Videre har jeg utforsket ulike teknikker innen naturlig språkbehandling (NLP) som kan brukes til å identifisere geografisk informasjon i nyhetsartikler. Fokuset var å overkomme utfordringene knyttet til bruk av NLP og tekstanalyse for å hente ut geospatiale relasjoner fra nyhetsinnhold.

Målet var å utvikle en fungerende prototype som kan ekstrahere og lagre geospatiale data, og gi innsikter gjennom kunnskapsgrafen som kan støtte journalister i deres arbeid. Resultatene av forskningen vil bidra til automatiserings-journalistikk og vise hvordan ny nyhetsrapportering kan utnytte avanserte teknologier som GPT-3/4 gjennom å utforske samspillet mellom NLP, geospatiale ontologier og journalistikk.

Innhold

Figurer

Tabeller

Kapittel 1

Introduksjon

1.1 Motivasjon

Det er et stort potensial for innovasjon innen automatiserings-journalistikk. Som en masterstudent interessert i tekstanalyse og utfordrende programmering, har det vært en glede å bidra til dette prosjektet ved å hente ut kunnskap fra nyhetsrapporter og andre kilder for å støtte journalister i deres arbeid. Jeg har utviklet en programvare for å hente ut relasjoner mellom steder nevnt i teksten, og sammenligne resultatene med et datasett som jeg har laget. Dette prosjektet har vært unik mulighet for meg å utvikle mine ferdigheter i tekstanalyse og naturlig språkbehandling, og å bidra innen automasjonsjournalistikk.

Bruken av automatisering innen journalistikk er et relevant og spennende fagområde å undersøke, med muligheten til å arbeide med avansert teknologi som GPT-3 og utforske samspillet mellom NLP, geospatiale ontologier og journalistikk. Utfallet av denne forskningen kan bidra til bedre løsning for automasjonsjournalistikk.

1.2 Problemforklaring

Jeg skal i min avhandling utvikle en programvare som kan ekstrahere geospatiale data fra nyhetstekster ved hjelp av GPT-3 og lagre det i en kunnskapsgraf. Det er en utfordring å trekke ut presis informasjon om steder, geografiske trekk og koordinater fra ustrukturerte tekster. Dette prosjektet vil derfor fokusere på å løse denne utfordringen ved å utvikle en programvare som kan utnytte GPT-3 til å ekstrahere relevant informasjon og lagre det i en kunnskapsgraf. Jeg vil

også undersøke ulike geospatiale ontologier som kan brukes til å identifisere relasjoner mellom dataene lagret i grafen. Det endelige målet med prosjektet er å utvikle en fungerende prototype som kan evalueres og diskuteres i forhold til de opprinnelige kravene og problemstillingene.

1.3 Oppgaver

- utvikle en programvare som har muligheten til å ekstrahere geospatiale data og lagre det i en kunnskapsgraf.
- Hente ut relasjoner mellom dataene i kunnskaps-grafen
- Prøve ut NLP teknikker og metoder som kan brukes til å idenfisere geografisk informasjon
- Finne ut av de største utfordingene med å bruke NLP og tekstanalyse til å hente ut geografiske relasjoner fra nyhetsinnhold, og hvordan de kan overkommes.

Kapittel 2

Bakgrunn

Det er mange muligheter for å utnytte avansert tekstanalyse og kunstig intelligens for å støtte journalister. Potensialet for å utnytte naturlig språkbehandling og geospatial ontologier for å hente ut geografisk kunnskap fra nyhetstekster er et slikt område. Bruken av GPT-3-teknologi for å hente ut denne informasjonen har vært mitt fokus, med målet om å identifisere hendelsessteder og relasjoner mellom dem.

2.1 ChatGPT

2.1.1 Hva er GPT-3?

Floridi and Chiriatti (2020) diskuterer rundt GPT-3, som er utviklet av organisasjonen OpenAI. OpenAI som har uttalt at de har som mål å promovere og utvikle vennlige kunstlig intelligens-artefakter som skal bidra til et bedre samfunn. GPT-3 er en autoregressiv språkmodell av tredje generasjon som bruker dyp læring for å produsere tekst som ligner på menneskelig skrevet tekst. Med andre ord, det er et beregningssystem som er designet for å generere sekvenser av ord, kode eller annen data ut ifra en gitt kildeinndata som kalles ”ledetekst”. Det brukes for eksempel i maskinoversettelse basert på statistisk informasjon.

2.1.2 Hva er store språkmodeller?

Store språkmodeller er en kategori av modeller trent på enorme mengder data, noe som gjør dem i stand til å tolke og generere naturlig språk og andre typer innhold for å utføre et bredt spekter av oppgaver.

2.1.3 Store språkmodeller hallisunerer

Bang et al. (2023), skrives det om store språkmodeller som ChatGPT sin evne til å utføre mange forskjellige type oppgaver, inkludert et begrep som heter TOD, eller "Task Oriented Dialogue". Når det kommer til å bruke denne språkmodellen, så har den et problem med at den har all sin data i ett spesifikt korpus av tekst. Dette hindrer modellen i å ta i mot nyere kunnskap, men også gjerne forklare hvorfor den svarer det den svarer. Dette leder da til mange hallusinasjoner, som da kan skape usikkerhet i svar som blir gitt, siden den ikke kan verifisere om det den svarer er korrekt. Hallusinasjoner referer til at modellen ikke har tilgangen til kunnskapen som trengs til å verifisere at denne informasjonen den gir deg stemmer. Et eksempel på dette er viss du ber den generere kilder gitt et spesifikt tema, så finner den på kildene, enten om de stemmer eller ikke. I en artikkel av Brown et al. (2020) hvor det blir forklart at når det kommer til å servere GPT-3 lengre tekster, så kan kvaliteten av dens respons synke en viss grad, og kan ende opp med å miste sammenhengen i generering av svar tekst, samt også motsi seg selv. GPT-3 sin output kan også inneholde hva som kalles for sequitur-setninger. En sequitur-setning er setninger som ikke gir logisk mening i forhold til forhold til tidligere nevnt tale/text. Dette kan eventuelt skape et problem med å hente ut spesifikk data fra et korpus av større tekst, siden den kan bli usikker og generere forventet resultat. Hvorfor er ideen om hallusinasjoner viktig for mitt prosjekt? Oppgaven går i hvordan GPT modellen bruker ledeteksten for å generere en tekst som inneholder de geospatielle dataene. Siden vi krever generering fra modellen, så er det viktig at det som kommer ut er korrekt.

2.1.4 Ekstrahering av informasjon

En relevant tillegg til dette fagområdet er bruken av semantiske teknologier som ontologier for å støtte geospatial data organisering og gjenfinning av informasjon fra geospatielle data. Dette åpner opp for spennende muligheter til å optimalisere AI- og tekstanalyse verktøy som GPT-3 for å hjelpe journalister med å finne og organisere nyhetsrapportering basert på geografiske hendelser Zhang et al. (2017).

2.1.5 Ledetekst-Ekstraksjon

Sharma et al. (2022), diskuterer hvordan ChatGPT kan brukes til å mate den informasjon, og ekstrahere spesifikke entiteter/deler fra teksten ved bruk av statistisk tekst analyse. Metoden deres gikk ut på å ha en unik måte å fullføre zero-shot ledetekster til å ekstrahere spesifikke deler av teksten. I dette prosjektet, så ble det fullført statistisk analyse på et datasett ved bruk av pre-prosessering, data-transformering, og selve analysen. Siden det brukes et datasett, og jeg bruker fritekst, så er det ikke alt i teksten som er like relevant til min oppgave. En nyhetsartikkel kan også kan bli lagt til i et datasett, eller finne datasett med nyhetsartikler fra før av, hvor eg da kan gjøre lignende pre-prosessering slik som behandling av dataene. Dette kan være alt fra å hente ut spesifikk data, eller filtrere ut data. Det ble konkludert i deres rapport at mesteparten av tiden, gir GPT3 veldig nøyaktige og rette svar. De fant også at viss GPT3 generer uten noe form for domene kunnskap, så har den nedsatt nøyaktighet i forhold til med domene kunnskap.

2.2 Geospatial ontologier

2.2.1 Hva er en geospatial ontologi?

En geospatiell ontologi gir en formalisert representasjon av geografiske enheter og relasjoner mellom dem på en måte som er forståelig for maskiner. Ontologien kan brukes til å løse semantiske problemer ved å etablere en felles forståelse av geografiske begreper, som geografiske egenskaper, geospatielle datatyper, koordinatsystemer, kartprojeksjoner og romlige relasjoner.

2.2.2 GeoClust & GeoCo

En viktig del av masterprosjektet mitt er å jobbe med geospatialle data og trekke ut denne informasjonen fra nyhetstekster, for så å representere det i en kunnskapsgraf. I en forskningsartikkel av Wang et al. (2010) ble et rammeverk som brukes for å klargjøre geospatiale data kalt GEO_CLUST nevnt, der det bygges opp på en geospatial ontologi kalt GeoCo ontologi, som er ment for å representere geospatial og domenekunnskap. Dette kunne være et ideelt emne å integrere på grunn av mitt prosjekt som krever lagring og representasjon av geospatiale data. Jeg nevnte tidligere at dataene ville bli lagret og representert inne i en kunnskaps-graf.

Det finnes måter du kan søke gjennom disse grafene for å få dataene du er ute etter, for eksempel spørrespråket SPARQL. Det er et spesifikt spørrespråk som er spesifikt for spørringer for geospatiale data. Dette språket er nevnt i en artikkel av Claramunt (2020), der de refererte til det som GEOSPARKQL. GEOSPARKQL-språket er en RDF SQL-basert spørring som manipulerer geospatiale RDF-data. GEOSPARQL er en utvidelse av spørrespråket SPARQL.

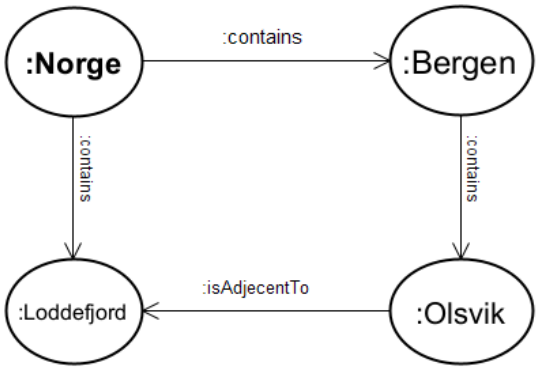
2.2.3 GEGIS rammeverket

Zhang et al. (2017) dokumenterer et lignende type prosjekt sammenlignet med prosjektet knyttet til mitt master prosjekt. De introduserer en geo-hendelsesbasert geospatial informasjonstjeneste der de skraper nettet for geospatiale data og lagrer det. Dette gjøres ved hjelp av et rammeverk de kaller GEGIS-rammeverket. GEGIS består av tre forskjellige moduler: sosial data service modul, geospatial servicemodul og service manager modul. Alle disse jobber sammen med å samle inn sosiale data som skal lagres, noe som er ideen bak mitt prosjekt. Verktøyene som brukes og teknikkene er forskjellige sammenlignet med forskningsartikkelen som forklarer GEGIS-rammeverket.

2.3 Kunnskaps-graf fremstilling

2.3.1 Hva er en kunnskaps-graf?

Ifølge IBM (2024a) er en kunnskapsgraf, også kalt et semantisk nettverk, er en måte å organisere informasjon på som viser forholdet mellom ulike elementer. Den består av noder, kanter og etiketter, der noder representerer konsepter eller objekter, kanter representerer forbindelsene mellom dem, og etiketter gir ytterligere informasjon om disse forbindelsene. For eksempel kan en node være en person eller et sted, en kant kan representere en relasjon som tilhørighet eller tilhørighet, og etiketten kan gi detaljer om denne relasjonen. Denne formen for strukturert kunnskap lar systemer forstå sammenhenger mellom ulike datapunkter og gi meningsfull informasjon basert på disse relasjonene. Denne relasjon strukturen kalles også en RDF. Et eksempel av en kunnskaps-graf kan du se i illustrasjon 2.1.



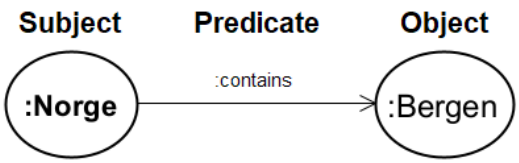
Figur 2.1: Diagram eksempel av en kunnskaps-graf.

2.3.2 Hva er en RDF?

RDF gir en metode for å uttrykke forhold mellom ulike data på en strukturert måte. I hjertet av RDF er ”triples”, som er uttalelser som består av et emne, et predikat og et objekt. Emnet representerer ressursen som beskrives, predikatet representerer egenskapen eller forholdet til ressursen, og objektet representerer verdien av den egenskapen eller forholdet. Dette blir mer forklart av W3C (2014).

For eksempel, i uttalelsen ”Bergen er i Norge” , er ”Bergen” emnet, ”er i” er predikatet, og ”Norge” er objektet. Denne enkle strukturen gjør at RDF kan representere et bredt spekter av informasjon på en måte som kan behandles og forstås lett av maskiner.

RDF brukes ofte som grunnlag for kunnskapsgrafer og andre semantiske webteknologier. Det gjør det mulig å koble data sammen på meningsfulle måter, slik at det blir enkelt å navigere og spørre store datasett. Et eksempel av en RDF kan du se i illustrasjon 2.2.



Figur 2.2: Diagram eksempel av en RDF.

2.3.3 Generering av GeoKG av 'multisource' data

Dataen som er samlet fra tekst analysen av nyhetstekster lagres i en kunnskaps-graf for å da bli representert. Grunnet dette, er det viktig å ha en bra og organisert måte å konstruere slike GeoKG'er (Geografiske Kunnskaps-Grafer). I en artikkel av Guo et al. (2021), blir det skrevet om en metode for å konstruere GeoKG'er utifra flerkildedata. Det som menes med flerkilde-data, er data som er samlet fra flere kilder. Det kan være sensorer, sosiale medier, nettsider, undersøkelser og mer. Det som er mest relevant for oss er den delen hvor vi ekstraherer data-en fra en nyhets-artikkel. Selv om delen av flerkildedata ikke er veldig relevant i forhold til prosjektet, så er utviklingen av grafen, og lignende veldig relevant.

Kapittel 3

Forskningsproblem

Jobben min var å ekstrahere geospatial data ved bruk av en form for geospatial ontologi, som for eksempel kunne vært GeoCo nevnt tidligere i teksten. Når denne dataen er samlet, skal den lagres i en kunnskaps-graf hvor den skal kunne representeres, og gjerne lages på en lignende måte som GEGIS. Her er hvor eventuelt GEOSPARQL kan bidra til å hente ut de nødvendige geospatiale dataene fra grafen. Ved alt dette er den interessant å finne ut av hvordan ChatGPT kan brukes til å hente ut geospatialle data og eventuelle negative og/eller positive sider ved det. Dataen skal sammenlignes med et "gull-standard" datasett som skal lages for å sammenligne med output fra modellen.

Dette leder da til forskningsspørsmålet, i tillegg til underspørsmål

RQ0: Hvilke NLP-teknikker og metoder kan brukes for å identifisere geografisk informasjon i nyhetsartikler?

RQ1: Hvordan kan funksjonalitet til ChatGPT utnyttes til å identifisere strukturert geografisk informasjon i nyhetsartikler.

RQ2: Hva er de største utfordringene knyttet til å bruke NLP og tekstanalyse til å hente ut geografiske relasjoner fra nyhetsinnhold, og hvordan kan disse utfordringene overvinnes?

Kapittel 4

Forskningsmetode

4.1 Design Science

Forskningsmetoden som kommer til å være sentralt for dette prosjektet er design science. I denne oppgaven skal det utvikles en artefakt som har muligheten til å ekstrahere geospatiale data fra en nyhetstekst ved hjelp av en stor språkmodell som ChatGPT/GPT-3/GPT-4. Deretter, skal dataene lagres i en kunnskaps-graf hvor relasjonene mellom disse skal kunne fremvises. Måten det programmet kommer til å fungerel er at brukeren først starter programmet. Deretter gir brukeren en input, som i dette tilfellet er en nyhetstekst. Når nyhetstekst input er gitt, begynner programmet ved hjelp av GPT å finne geospatiale data i teksten. Etter alle geospatiale dataene er funnet, så lagres alle som tripler i en kunnskaps-graf. Deretter kan brukeren be om å hente relasjoner fra den geografiske kunnskaps-grafen mellom de forskjellige geospatiale dataene. Her er ett til diagram hvor det viser på hvilke nivå denne artefakten skal bygges, evalueres. teoretiseres, og bekreftes:

Tabell 4.1: March og Smith tabell av hvilke nivåer webapplikasjonen bygges, evalueres, teoretiseres, og bekreftes.

	Bygge	Evaluere	Teoretisere	Bekrefte
Konstrukt	X	X		
Modell	X	X		
Metode	X	X		
Instans				

Diagram som viser et forsknings rammeverk tilpasset til mitt prosjekt, basert på en forsk-

ningsartikkel av March and Smith (1995).

4.1.1 Hva er March og Smith diagrammet?

Diagrammet illustrerer hvordan et program går gjennom ulike stadier, hver assosiert med ulike nivåer. Disse stadiene representeres av fire nøkkelkonsepter: Konstrukt, Modell, Metode og Realisering. Forklaringene nedenfor er hentet fra March and Smith (1995).

1. Construct
2. Model
3. Method
4. Instantation

Construct På det grunnleggende nivået representerer konstrukten de underliggende prinsippene, teoriene og konseptuelle rammeverkene som programmet er bygget på. Den inneholder de grunnleggende ideene og målene som veileder designprosessen.

Model Beveger seg bort fra det abstrakte, involverer modellstadiet utviklingen av konkrete representasjoner eller prototyper basert på de etablerte konstruktene. Disse modellene fungerer som håndfaste implementeringer av de teoretiske konseptene, slik at tredje parter kan utforske praktiske implikasjoner og gjennomførbarhet.

Method Når programmet tar form, omfatter metodestadiet strategiene, teknikkene og prosedyrene som brukes til å implementere og evaluere modellene i virkelige settinger. Denne fasen innebærer grundig eksperimentering, datainnsamling og analyse for å vurdere effektiviteten og ytelsen til de designede løsningene.

Instantation Til slutt, innebærer realiseringsstadiet utrulling og integrasjon av de utviklede løsningene i praktiske kontekster eller systemer. Det fokuserer på den faktiske implementeringen av de designede artefaktene, med hensyn til faktorer som brukervennlighet, skalerbarhet og tilpasningsevne til ulike miljøer.

4.1.2 Forklaring av diagrammet

Bygge Artefakten som skal utvikles på et konstrukt og metode nivå. På konstrukt nivå skal det høyt sannsynlig utvikles en geospatiale ontologi som jeg skal bruke til å hente ut den geofrafiske datane fra nyhetstekster. Artefaken bygges på et metode nivå på grunn av at jeg skal lage metoden som tar i bruk OpenAI sin API til å hente ut dataene, lagre de i en kunnskapsgraf of visualisere og ekstrahere relasjonene. Prosjektet bygges også på et modell nivå, siden det jeg fullfører fin-justering av allere eksisterende GPT modeller for å forbedre nøyaktighet for mine analyser.

Evaluere Når det kommer til hvilke nivå denne prosjektet skal evalueres på, så blir det på tre nivåer, og de er: Konstrukt, Modell, og Metode. Grunnen til at konstrukt nivå er innkludert, er siden jeg skal utvikle en geospatial ontologi, og evaluere denne ontologien, som ligger på et konstrukt nivå. Jeg skal også evaluere selve artefakten jeg har utviklet, på en modell nivå, og et metode nivå. Instans nivå blir ikke tatt i bruk i denne sammenhengen, på grunn av at denne løsning skal ikke bli implementert i en bedrift, hvor den skal evalueres.

Teoritisere De fleste delene av prosjektet skal teoretiseres på en eller annen måte, men er langt ifra den største delen av prosjektet, som er grunnen til at den ikke er innkludert i diagrammet. For å lage en effektiv geospatiale ontologi, så burde det være en grunnmur av teori rundt hvordan både ontologier generelt fungerer og hvordan dette kan kobles til en kunnskapsgraf. Derfor blir det teoretisert på konstrukt-nivå på grunn av ontologien. Prosjektet teoretiseres også på et metode nivå, med at dataen fra ontologien skal kobles sammen med kunnskapsgrafen, og deretter hente ut relasjonene via et spørrespråk, slik som SPARQL eller GEOSPARQL.

Bekrefte Det jeg skal bevise er hvordan geospatiale data kan ekstraheres og lagres i kunnskapsgraf og deretter vise relasjoner mellom geospatial data. Jeg skal bekrefte om den geospatiale ontologien fungerer som forventet. Skal bekrefte om ChatGPT klarer å hente ut geospatiale data ved hjelp av ontologien, dette er modell nivået. Skal også bekrefte om min metode klarer å lagre dataen i en kunnskapsgraf og visualisere og ekstrahere relasjonene mellom dataene.

4.2 Eksperimentelt design

Designet av selve oppgaven er å utvikle og teste ett system på nyhetsartikler, I en slik oppgave så kan det være at en slik analyse ville være uegnet. Dette er fordi at det gjerne ikke er en klar brukscase eller anvendelse for denne analysen ettersom den er den er kvantitativ og fokuserer på statistisk analyse og maskin-læring. Videre kan det være etiske og personvern-relaterte bekymringer knyttet til innsamling og behandling av personlig informasjon, som ble nevnt tidligere i kapittel 4.

4.2.1 Trusler mot validitet

En potensiell trussel er gyldigheten til geospatiale data som blir hentet ut fra nyhetsartiklene, noe som blir diskutert i denne artikkelen av Balakrishnan (2019). I denne artikkelen diskuteres det om unøyaktigheten i geospatial data og valideringsteknikker for denne dataen. Selv om programvaren kan være effektiv i å identifisere og hente ut geospatialle data, er det alltid muligheten for unøyaktigheter eller feil i dataene på grunn av kompleksitetene i naturlig språkbehandling og nyansene i språket. I tillegg kan utvalget av nyhetsartikler som brukes i analysen, ikke være representativt for den bredere populasjonen av nyhetsartikler, noe som kan introdusere utvalgsbias og true den eksterne validiteten til resultatene.

4.2.2 Etiske problemstillinger

Personvern Personvern er en viktig del av ethvert system som håndterer sensitive data. Når det gjelder et system for å trekke ut geospatialle data fra nyhetstekst, er det viktig å sørge for at personopplysninger blir behandlet på en sikker og konfidensiell måte. For det første må systemet ha mekanismer for å beskytte personvernet til personer som er nevnt i nyhetstekstene. Dette kan inkludere å fjerne eller anonymisere navn og andre personlige opplysninger som kan knyttes til en bestemt person. Dette kan gjøres ved hjelp av maskinlæringsteknikker for å identifisere og anonymisere personopplysninger. For det andre er det viktig å sørge for at dataene som trekkes ut fra nyhetsteksten, ikke brukes til å identifisere enkeltpersoner. Dette kan oppnås ved å aggregere dataene på en måte som gjør det umulig å trekke ut individuelle data. For eksempel kan dataene grupperes etter land eller region i stedet for å være knyttet til en bestemt person eller adresse. Til slutt er det viktig å sørge for at systemet følger gjeldende personvern-

lover og -regler, for eksempel personvernforordningen (GDPR) i EU The European Union and the Council of the European Union (2016). Dette kan inkludere å sørge for at personopplysninger blir lagret på en sikker måte, at brukerne får tilstrekkelig informasjon om hvordan deres data blir brukt, og at de har rett til å be om at deres data blir slettet eller endret hvis de ønsker det. I mitt tilfelle blir det ikke lagret personlig identifiserbar informasjon i nyhets-databasen.

Nøyaktighet i Geospatial Data Nøyaktighet er en viktig faktor å vurdere når det gjelder programvare for utvinning av geospatiale data fra nyhetstekst og opprettelse av en kunnskapsgraf. For å sikre nøyaktighet, må programvaren kunne gjenkjenne og trekke ut relevant informasjon fra teksten på en presis måte. Dette kan inkludere å identifisere og utvinne navn på steder, koordinater, datoer og andre relevante detaljer. En annen viktig faktor for å sikre nøyaktighet er å validere og bekrefte den utvinne geospatiale informasjonen. Dette kan oppnås ved å koble den utvinne informasjonen til pålitelige kilder og datasett som allerede er verifisert, eller ved å bruke algoritmer og teknikker for å verifisere dataene på egen hånd. Det er også viktig å forstå begrensningene i programvaren og dens evne til å trekke ut nøyaktig informasjon fra ulike typer tekster og på forskjellige språk. Dette kan kreve ytterligere tilpasning og testing av programvaren for å sikre at den fungerer optimalt i ulike miljøer og kontekster. Generelt sett er det viktig å sikre at programvaren opererer etisk og ansvarlig for å fremme bruken av geospatiale data til positive formål, samtidig som man unngår negative konsekvenser.

Validering Av Data Generelt sett er det viktig å sørge for at programvaren fungerer etisk og ansvarlig for å fremme bruken av geospatiale data til positive formål, samtidig som man unngår eventuelle negative konsekvenser. Vi må også være oppmerksomme på at kvantitativ datainnsamling kan ha visse begrensninger og at det er viktig å validere vår tilnærming for å sikre nøyaktigheten av resultatene. Det er derfor viktig å nøye vurdere eventuelle begrensninger og utfordringer som kan påvirke gyldigheten av mine funn.

Sikkerhet

Når man utvikler en webapplikasjon som involverer å ekstrahere geolokasjonsdata, så er det en del sikkerhetsvurderinger som må bli tatt hensyn til.

Bruk av et Governance Framework For å håndtere personvern- og sikkerhetsimplikasjoner, er det viktig å utvikle et eierstyrings rammeverk. I forhold til mitt prosjekt, så blir dette rammeverket å finnne ut en strategi for å implementere geo lokasjon i min applikasjon. Denne strategien burde være i forhold til potensielle personvern og sikkerhets standarer for å beskytte personlig data som eventuelt dukker opp i nyhetstekstene som blir brukt til geo lokasjon ekstraksjon. Dette kan eventuelt være å bruke plassholdere istedenfor navn etter datainnsamling for enkelt personer. Alle som bruker systemer og er innvolvert i datainnsamlng bør bli trent opp til å opprettholde disse prosyderene for å beskytte dataene som blir samlet. Hvordan et eierstyre rammeverk fungerer i forhold til geolokasjon er forklart mer i dybde i en artikkel av Estes (2016).

4.3 Metoder for evaluering

I mitt studiet utnyttes offline evaluering som metodikk for å vurdere effektiviteten og kvaliteten på artefaktet utviklet ved bruk av OpenAI API for geospatial datauthenting. Offline evaluering innebærer en sammenligning av forhåndsdefinerte data tabeller mot output generert av artefaktet, for å vurdere dens dyktighet i utvalgte oppgaver uten behovet for behandling av sanntidsdata eller implementering i live-scenarier. Denne tilnermingen er særlig passende for analysen av geospatial data, der nøyaktighet, pålitelighet og kontektspesifikk ytelse er av størst viktighet. Denne analysen blir fullført på GPT-3-Turbo og GPT-4-Turbo for å sammenligne de to modellene i forhold til ytelse og kvalitet.

I evalueringen av nøyaktighet til svaret fra API'et tar metodene hensyn til de innbygde kompleksitetene i strukturerte datarepresentasjoner. Tradisjonelle tilnærminger til nøyaktighetsvurdering baserer seg ofte på binære sammenligninger, og overser subtile variasjoner i datastrukturen som kan oppstå i virkelige scenarioer. Derfor foreslår jeg omfattende metodologi som tar hensyn til flere faktorer og bruker nyansert scoringsform for å gi en mer informativ måling av nøyaktighet.

Vurdering av strukturelle komponenter Et viktig aspekt ved min metode er hensynet til strukturelle komponenter innenfor JSON-objekter. Jeg erkjenner at JSON-objektet kan va-

riere i sammensetning, med ulikt antall enheter som for eksempel steder og relasjoner. Ved å evaluere tilstedeværelsen av korrektheten til hver komponent, tilpasser min tilnærming seg variabiliteten i datastrukturen og sikrer en helhetlig vurdering av nøyaktigheten.

Scoringsmekanisme Min metode inkluderer en scoringsmekanisme som tildeler vekter til ulike komponenter basert på deres betydning. For eksempel kan korrekte relasjoner anses som viktigere enn det nøyaktige antall steder. Ved å tildele passende vekter prioriterer vi evalueringen av komponenter som er mest relevante for den underliggende datadomenet, og dermed forbedret jeg nøyaktighetsvurderingsprosessen.

Håndtering av variabilitet En annen styrke ved min tilnærming ligger i dens evne til å håndtere variasjoner i forutsagte JSON-objekter. Vi erkjenner at forsutsagte resultater kan inneholde ekstra eller manglende informasjon sammenlignet med den faktiske sannheten. For å håndtere dette, straffer vi får både avvik forholdsmessig, og sikrer dermed en rettferdig og balansert evaluering av nøyaktigheten uavhengig av strukturelle forksjeller mellom forutsagte og faktiske objekter.

Interpretasjonsevne Gjennomsiktighet og interpretasjonsevne er avgjørende i enhver evaluering. Derfor gir vår metode en klar oppdeling av nøyaktigheten i individuelle komponenter, som for eksempel stednøyaktighet og relasjonsnøyaktighet. Dette gjør det mulig for interessenter å få innsikt i styrkene og svakhetene til prediksjonssystemet, og legger til rette for informerte beslutninger og videre optimaliseringsinnsats.

Oppsummert tilbyr min foreslåtte metode en omfattende og nyansert tilnærming til å evaluere nøyaktigheten til forutsagte JSON-objekter. Ved å vurdere flere faktorer, bruke en sofistikert scoringmekanisme og prioritere tolkningsvennlighet, forbedrer vår metode påliteligheten og relevansen til nøyaktighetsvurderinger i datastyrte applikasjoner.

Kapittel 5

Forsknings plan

5.1 Ressurser

For å fullføre dette prosjektet benyttet jeg meg av flere ressurser. Disse inkluderte en personlig datamaskin, Python programmeringsspråk, og ChatGPT. ChatGPT spilte en nøkkelrolle i å trekke ut geospatiale data fra tekstene som ble brukt til å bygge kunnskapsgrafene. Programmeringsspråkene Python ble brukt til å utvikle programvaren og implementere kunnskapsgrafene. IDE-er ble også brukt for å optimalisere utviklingsprosessen og sørge for effektiv kode. Disse ressursene var avgjørende for å fullføre prosjektet på en tilfredsstillende måte.

5.2 Risikoer

Det var noen risikoer knyttet til mitt prosjekt. En av de største risikoene var at GPT-3 vill ikke være nøyaktiv nok til å ekstrahere geospatiale data fra nyhetstekster på en konsistent måte, dette ble også snakket mer i dybden om i en artikkel av Olmo et al. (2021). For å motvirke dette sørget jeg for å teste og evaluere resultatene nøye, og gjorde justeringer i programvaren og ontologien når det var nødvendig. En annen risiko var at det kunne oppstå etiske problemer knyttet til bruk av personopplysninger og personvern. Jeg sørget for å følge alle relevante retningslinjer og regulering for personvern og datasikkerhet, og begrenset bruken av personlig informasjon til det som var nødvendig for å ekstrahere geospatiale data. Jeg tok også hensyn til potensielle etiske utfordringer knyttet til kunstig intelligens og automatisering. Jeg sørget for å utføre grundig etisk vurdering av prosjektet, og sørget for at det ikke forårsaket skade eller uønskede konsekvenser for enkeltpersoner eller samfunnet som helhet. Eksempel på dette er

anonymiseringen av enkeltpersoner i nyhets-databasen jeg har samlet opp. Hvis det skulle oppstå problemer eller bekymringer, var jeg åpen og transparent om disse og søkte råd og veiledning fra relevante experter og interessenter.

5.3 Etiske Problemstillinger

Det var en del etiske hensyn som kunne oppstå i et slikt prosjekt som dette. Først og fremst var det viktig å beskytte personvernet til personene som var involvert i nyhetsartiklene som ble brukt til modellen som skulle utvikles. Det var også viktig å være bevisst på eventuelle fordommer eller skjevheter som kunne eksistere i dataene som skulle brukes, og å jobbe aktivt for å motvirke disse, dette ble diskutert i en forskningsartikkel av Coeckelbergh (2019). Videre kunne det være etiske spørsmål knyttet til bruken av kunstig intelligens i journalistikk, spesielt når det gjaldt automatisering av arbeidsoppgaver som tidligere ble utført av mennesker. Det var viktig å sørge for at modellen ikke ble brukt til å spre falske nyheter eller propaganda, og å være åpen om begrensingene eller usikkerhetene ved modellen. Dette ble snakket mer om i en forskningsartikkel av Zuiderwijk et al. (2021) hvor det ble reflektert rundt potensiell misbruk av en potensiell kunstig intelligens modell.

Kapittel 6

Utvikling

6.1 Geopatielle uttrekk ved bruk av ledetekst-utvikling

6.1.1 Hva er ledetekst utvikling?

Ledetekst utvikling er en prosess som har vokst frem som en nøkkelkomponent i utviklingen av generative kunstige intelligenssystemer. I hjertet av denne prosessen ligger ideen om å veilede AI til å produsere ønskede resultater ved å gi den detaljerte instruksjoner. I denne prosessen er valget av format, fraser, ord og symboler avgjørende for å veilede AI til å samhandle på en mer meningsfull måte med brukerne. Det krever en blanding av kreativitet og prøving og feiling for å utvikle et sett med inngangstekster som kan sikre at en applikasjons generative AI fungerer som forventet. En ledetekst er en naturlig språktekst som ber den generative AI om å utføre en spesifikk oppgave. Generativ AI er en type kunstig intelligens som er i stand til å skape nytt innhold, for eksempel historier, samtaler, videoer, bilder og musikk. Det er viktig å merke seg at ikke alle typer ledetekster genererer nyttig svar. Generative AI-systemer krever kontekst og detaljert informasjon for å produsere nøyaktige og relevante svar. Ved å designe ledetekster på en systematisk måte, kan vi oppnå mer meningsfulle og brukbare svar, noe som blir forklart av Amazon (2024). I mitt tilfelle, bruker jeg ledetekst utvikling til å ekstrahere geospatial data fra nyhetstekster.

6.1.2 Lokasjon og relasjon uttrekk

For å hente ut geografisk informasjon fra nyhetstekster, bruker jeg ledetekst-utvikling til å instruere AI-modellen om å søke etter steder og relasjoner i teksten. Spørsmålet som sendes til AI-modellen beskriver en bestemt geospatial ontologi som modellen skal følge når den ekstraherer geografiske data. Denne ontologien er nærmere forklart i kapittel 6.3. I spørsmålet bruker jeg en spørreteknikk som kalles Few-Shot læring, som er litt forskjellig fra Zero-Shot learning som ble presentert i kapittel 2.1.5. Forskjellen mellom Few-Shot og Zero-Shot er at jeg gir eksempler på hvordan svaret på spørsmålet skal formateres. I første utkast av sted- og relasjonsutvinning, forklarte jeg AI-modellen at svaret på spørsmålet skulle formateres som et JSON-objekt, og dette gjorde jeg med flere eksempler på hvordan JSON-objekter ville se ut i den gitte konteksten. Hvis bare ett eksempel blir gitt, blir spørsmålet klassifisert som One-Shot læring, og ikke Few-Shot. Forskjellen mellom Few-Shot og One-Shot er mer detaljert i et preprint av Tyukin et al. (2021).

6.2 Utviklingen av Webapplikasjonen

6.2.1 Brukergrensesnitt

Grafvisning

Brukergrensesnittet blir en side hvor både kunnskapsgrafen og relasjonsekstraksjonen blir visualisert. For grafbildet så blir det i form av en HTML-fil som blir plassert i en HTML-iframe for å integrere den i webapplikasjonen. HTML-filen blir generert av en Python-pakke som heter PyVis, som er en pakke for visualisering av mange former for grafer, men som er spesielt effektiv for å visualisere kunnskapsgrafer. Her kan du se alle nodene som er geolokasjoner og kantene som er relasjonene mellom dem. Relasjonsekstraksjonen blir vist som en tabell under grafbildet, hvor du kan se hvilke geolokasjoner som har hvilke typer relasjoner, basert på tekstanalysen. Dette kan hjelpe deg med å forstå sammenhengen mellom ulike steder og hendelser i teksten. Du kan også klikke på en node eller en kant i grafbildet for å få mer informasjon om den.

Relasjonsvisning

Når det kommer til fremvisning av relasjonene mellom geo-lokasjonene, så er det en interaktiv meny som brukere av webapplikasjonen kan bruke til å velge en geolokasjon. Etter valget geolokasjon så får brukeren valge om å se alle geo-lokasjonene som denne noden har relasjoner til, samt hvilke relasjon som er valgt. Disse relasjonene skal bli vist som en strukturert liste som brukeren har tilgjengelig til å oppdatere med å spørre om nye relasjoner for å bli uthentet. Når nye relasjoner blir uthentet, så blir de gamle fjernet.

6.2.2 Interaktivitet

Interaktivitet er en vesentlig komponent i denne webapplikasjonen. Når det gjelder visning av kunnskapsgrafen, gir applikasjonen brukerne mulighet til å navigere fleksibelt rundt i grafen. Brukerne kan velge forskjellige noder og fremheve alle relasjoner og noder som er koblet til den valgte geolokasjonen (noden). Brukerne har også muligheten til å sende inn input for å oppdatere den viste grafen. Dette gir brukerne en dynamisk og interaktiv opplevelse, da de kan påvirke og se endringene i sanntid. Videre har brukerne av webapplikasjonen evnen til å hente ut relasjoner relatert til en eller flere spesifikke geolokasjoner. For eksempel, hvis byen Bergen blir valgt, vil alle geolokasjoner relatert til Bergen i grafen bli hentet ut og presentert for brukeren på en strukturert og forståelig måte. Dette gir brukerne en dypere forståelse av de komplekse relasjonene mellom forskjellige geolokasjoner, og muliggjør mer informerte og innsiktsfulle analyser. Samlet sett, ved å kombinere interaktivitet med kraftige dataanalyseverktøy, gir denne webapplikasjonen brukerne en unik og engasjerende måte å utforske og forstå geospatial data på.

6.2.3 Backend

Backend delen av webapplikasjon er ansvarlig for å ekstrahere geospatial data fra nyhetstekster ved bruk av OpenAI sitt API, hvor dataen blir lagt inn i kunnskapsgrafer. Backend av webapplikasjonen tar også i bruk av SPARQL og GEOSPARQL til å hente ut relasjonene mellom geospatial data. Backend delen av webapplikasjonen er implementert ved hjelp av Dash rammeverket som er bygget for å legge ut interactive web applikasjoner. Dette er alle komponentene som bygger opp backended.

- et API komponent, som bruker OpenAI sitt API til å utføre naturlig språk prosessering på tekst input. OpenAI API'et bruker både GPT-3 og GPT-4 til å ekstrahere entiteter og attributter fra tekster, som navn og koordinater. I mitt tilfelle er dette geo-lokasjoner og relasjoner mellom de.

- Et kunnskapsgraf komponent som bruker Python pakken Rdflib til å manipulere kunnskapsgrafer. Kunnskapsgraf komponentet konverterer JSON objekter fra OpenAI komponentet til RDF tripler. Kunnskapsgraf komponentet lagerer RDF triplene i en grafdatabase i form av en Turtle fil (.ttl).

- Et spørringskomponent, som bruker RDFlib til å utføre GEOSPARQL og SPARQL spørringer på kunnskapsgrafen. Komponentet tar imot spørringen fra frontenden og bruker Rdflib til å parse og evaluere dem. Spørringen blir da returnert tilbake til frontenden i form av et JSON objekt:

- et Dash komponent. Her bruker vi Dash rammeverket til å lage og opprettholde webapplikasjonen. Dash komponentet er ansvarlig for definere alle oppsettene og kall til applikasjonen. Dash bruker HTML og andre Dash hovedkomponenter til å lage grensesnittet til applikasjonen.

6.2.4 Håndtering av SPARQL og GEOSPARQL forespørsler

En del av webapplikasjonen er å hente ut relasjonene mellom de forskjellige geolokasjonene i grafen som er generert. SPARQL er et spørrespråk for å legge til, fjerne og hente ut data fra RDF-databaser. W3C (2024a) dokumenterer at SPARQL er et veldig fleksibelt språk og kan hente ut grafrelatert data. Det finnes flere språk som ikke er helt like SPARQL, men det finnes også spørrespråk som er bygget på SPARQL eller lagt til som en utvidelse. Det finnes språk som er spesifikt laget for spesifikke typer grafdata. GEOSPARQL er et spørrespråk som er bygget på SPARQL og som er utviklet for å håndtere geografisk/geospatial data. Du kan lese mer om GEOSPARQL i en tekst av Consortium (2024). GEOSPARQL kan brukes til å ekstrahere geospatiale attributter fra grafen. Et eksempel er at om man vil ekstrahere alle geolokasjoner i grafen som er innenfor 50 km geometrisk rekkevidde. GEOSPARQL og SPARQL vil bli brukt for å hente ut relasjonene som er samlet inn i kunnskapsgrafen.

I tillegg til å hente ut relasjonene, kan man også bruke GEOSPARQL og SPARQL til å utføre ulike operasjoner og analyser på grafdataene. For eksempel kan man bruke GEOSPARQL til å beregne avstander, arealer, buffersoner, kryssninger, og andre geospatiale funksjoner på geolokasjonene. Man kan også bruke SPARQL til å filtrere, sortere, gruppere, aggregere, og sammenligne dataene basert på ulike kriterier. Man kan også bruke SPARQL til å lage komplekse spørringer som involverer flere grafer, variabler, mønstre, og betingelser. Ved å bruke GEOSPARQL og SPARQL kan man få et bedre innblikk i grafdataene og de geospatiale relasjonene som finnes i dem.

6.2.5 Data lagring

For data brukt av webapplikasjon ble dette lagret som hovedsakelig to forskjellige format. For data som ble brukt av OpenAI API for å ekstrahere geolokasjon data ble lagret som CSV (Comma-Seperated-Values) filer. Dette var fordi at CSV filer kunne på en veldig simpel måte bli iterert gjennom på en strukturert måte for spesifiere hvilke tekst OpenAI skulle lete gjennom. I mitt tilfelle, ble dette for eksempel at vi letet gjennom nyhetsinnholdet, og ikke tittel. CSV filer var filer som var for lagring av data som var separert med komma. Med CSV filer kunne du lagre dataen i forhold til kategori navn, eller kolonne navn. CSV filer ble forklart mer i dybden av Hoffman (2018). Data som ble brukt til å evaluere hvor bra API'et til OpenAI utførte oppgaven ble også lagret i form av en CSV fil, for å så bli gjort maskinlæring på for å evaluere dette. RDF data som ble laget av webapplikasjon mens den kjørte ble lagret som en Turtle fil. Ttl sto for "Terse RDF Triple Language" og var et filformat brukt til å lagre RDF data. W3C (2024b) dokumenterer mer informasjon om Turtle filer. Om webapplikasjonen hadde kjørt tidligere, så hentet programmet denne RDF dataen fra filen, og la til ny data og lagret dette i samme fil.

6.2.6 Lagring av kunnskapsgraf

I utviklingen webapplikasjonen, ble kglab pakken i Python brukt for å bygge og manipulere kunnskapsgrafer. Dette tillot effektiv representasjon av komplekse relasjoner mellom data.

Opprettelse av kunnskapsgrafen

Etter opprettelsen av grafen, ble tripler lagt for å representere geolokasjonsdata. Hver triple representerer en relasjon mellom to noder i grafen. For eksempel, ble RDF brukt til å legge til geolokasjonsdata til grafen på følgende måte: (location_uri, rdflib.RDF.type, LOC.Bergen). Her representerer location_uri subjektet, rdflib.RDF.type representerer predikatet, og LOC.Bergen representerer objektet.

Lagring av Kunnskapsgraf

Etter at alle triplene ble lagt til i grafen, ble kunnskapsgrafen lagret til en fil. kglab støtter flere formater for lagring av grafer, inkludert i Turtle *Turtle*, N-Triples *.nt* og JSON-LD *.jsonld*. I dette prosjektet, ble kunnslapsgrafen lagret som en Turtle fil. Ved å bruke kglab i Python, ble det mulig å effektivt bygge og lagre en kunnskapsgraf som representerer geolokasjonsdata. Dette ka en kraftig måte å strukturere og analysere data på. I tillegg, ved å lagre grafen til en fil, ble det mulig hente tilbake dataene fra RDF'en og bruke til senere analyse.

6.3 Bruk av Geolokasjonsontologi

I webapplikasjonen skal det utføres spørringer mot OpenAI sitt API for både GPT-3 og GPT-4. For at denne ledeteksten skal være så effektiv som mulig, er det av stor betydning å tydeliggjøre hva GPT skal gjøre og hvilke forventninger du som bruker har til svaret. En dypere forklaring på dette konseptet vil bli presentert i kapittelet om ledetekst-utvikling. Siden webapplikasjonen skal integrere den geospatiale daten i en kunnskapsgraf, blir det essensielt at dataene som genereres av GPT presenteres på en strukturert måte. Dette vil gjøre det enklere å innlemme informasjonen i kunnskapsgrafen. Den påfølgende delen av ledeteksten er der ontologien spesifiseres. Ontologien blir konstruert på en måte som gjør den lett-forståelig i form av en graf når den deles med andre brukere. Et illustrerende eksempel kan være situasjonen der to ulike geografiske lokasjoner blir tolket og generert med en tilfeldig relasjon av GPT. Dersom relasjonen mellom disse lokasjonene ikke er klart definert, kan det føre til misforståelser i videre analyse av dataen. Derfor er det kritisk å sikre at ontologien gir et presist bilde av relasjonene mellom dataelementene, for å unngå feilaktige tolkninger og feilanalyser.

6.3.1 Introduksjon til ontologier

En kort forklaring på en ontologi i forhold til mitt prosjekt er representasjon av data på en måte som gjør den lett-forståelig for både personer og datamaskiner. Denne dataen består gjerne av enkelte konsepter og relasjonene mellom de. Forklaringen på geospatial ontologi, som er den formen jeg har brukt, blir forklart mer detaljert i kapittel 2.2 om geospatial ontologier.

6.3.2 Struktuering av geo data

Den geospatiale ontologien er en måte å representere geografiske data og relasjoner mellom dem på en standardisert og konsistent måte. Ontologien består av to hovedkomponenter: lokasjoner or relasjoner. Lokasjoner er geografiske enheter som kan ha ulike egenskaper som navn og koordinater. Relasjoner er forbindelser mellom to eller flere lokasjoner som beskriver hvordan de er relatert til hverandre, som avstand, retning, tilhørlighet, konflikt, samarbeid, osv.

For å gjøre det mulig for GPT å generere og tolke geospatiale data på en effektiv måte, er det nødvendig å definere en klar og entydig syntakt for ontologien. Syntaksen er regelsett som bestemmer hvordan dataen skal skrives og leses. I dette tilfelle er syntaksen basert på JSON-formatet, som er et popilært og lettleselig format for å lagre og utveksle data. JSON står for JavaScript Object Notation, og bruker nøkkel-verdi-par for å representere dataojekter. Et nøkkel-verdi-par består av en nøkkel, som er en streng som identifiserer en egenskap, og en verdi, som er dataen som tilhører den egenskapen.

Den geospatiale ontologien bruker JSON-formatet for å strukturere dataen i to objekter: et objekt for lokasjoner, og et objekt for relasjoner. Hvert objekt har en nøkkel som angir hva slags data den inneholder, og en verdi som er ett array av objekter som representerer de invididuelle dataelementene. Hvert dataelement har en unik identifikator som brukes til å referere til det i relasjonene. Hvert dataelement har også en egenskap til å beskrive det, som navn. Egenskapene kan variere avhengig av hva slags dataelement det er. For eksempel kan en lokasjon ha en egenskap som angir landet den tilhører, mens en relasjon kan ha en egenskap som angir styrken eller viktigheten av forbindelsen.

6.3.3 Integrering med kunnskapsgraf

Denne geospatiale ontologien passer godt for å innlemme geospatiale data i en kunnskapsgraf, fordi den har en klar og konsistent syntaks som gjør det enkelt å konvertere dataen i JSON-format til graf-format. Hvert dataelement i ontologien har en unik identifikator som kan brukes som en node i kunnskapsgrafen. Hver egenskap og verdi i dataelementetkan brukes som en egenskap og verdi i noden. Hver relasjon i ontologien har en kilde, et mål, en type og en verdi som kan brukes som en kant i kunnskapsgrafen. Kilden og målet er identifikatorene til de to dataelementene som er relatert, mens typen og verdien er egenskapene og verdiene i kanten.

Ved å bruke denne geospatiale ontologien til å innlemme geospatiale data i en kunnskapsgraf, kan man dra nytte av de mange fordelene som en kunnskapsgraf tilbyr. Noen eksempler er:

- Utføre komplekse spørringer på geospatiale data ved å bruke graf-algoritmer og -operasjoner, som korteste vei, nærmeste nabo, klyngedannelse, osv.
- Oppdage nye og skjulte mønstre, sammenhenger og innsikter i geospatiale data ved å bruke graf-analyse og -visualisering, som sentralitet, fellesskap, nettverksstruktur, osv.
- Forbedre kvaliteten og nøyaktigheten av geospatiale data ved å bruke graf-validasjon og -berikelse, som konsistenssjekk, duplikatfjerning, osv.

Spørringer og forbedring med kvalitet og nøyaktighet er spesielt hvilke i mitt tilfelle på grunn av lokasjonene uthentet burde være faktiske lokasjoner og av god visualiserings kvalitet.

6.3.4 Oppbygging av datasettet

En sentral del av offline evaluering er konstruksjonen av et omfattende testdatasett. Dette datasettet er nøye laget for å innklude et bredt utvalg av geografisk relevante lokasjoner sammen med deres korrekte relasjonskartlegging. Hver oppføring i datasettet er manuell klassifisert for å sikre høyest grad av nøyaktighet og relevans, som reflekterer virkelighetens geospatiale kompleksiteter og forhold. Formatet på datasettet speiler datastrukturen og formatet returnet av OpenAI API, noe som legger til rette for en direkte og meningsfull sammenligning mellom forventet utfall og faktisk API ytelse. Dette parallellen er avgjørende for å evaluere API'ets evne til nøyaktig tolkning og behandling av geospatial informasjon i henhold til brukerdefinerte krav.

6.3.5 Algoritme for vurdering av lokasjon og relasjonsnøyaktighet

Formålet med denne algoritmen er å evaluere nøyaktigheten til prediksjoner i mitt datasett, spesielt med fokus på lokasjoner og relasjoner.

Her er de forskjellige trinnene i algoritmen:

1. Datainnsamling og forberedelse
    - Innledningsvis samler jeg inn datasettet som inneholder både forutsagte og faktiske verdier.
    - deretter forbereder jeg dataene ved å sørge for at de er i riktig format og klar til bruk for evalueringen.
2. Beregning av lokasjonsnøyaktighet
    - Jeg starter med å sammenligne antallet og typene lokasjoner som er forutsagt med de faktiske lokasjonene i datasettet.
    - Dette gjøres ved å telle antallet forutsagte lokasjoner og antallet faktiske relasjoner, og deretter beregne en proposjonal score basert på dette forholdet
    - Lokasjonsnøyaktigheten gir en indikasjon på hvor godt modellen klarer å forutsi riktige lokasjoner.
3. Beregning av relasjonsnøyaktighet
    - Deretter fortsetter jeg på å vurdere nøyaktigheten av relasjonene mellom ulike lokasjoner i datasettet.
    - Dette inkluderer å sammenligne de forutsagte relasjonene med de faktiske relasjonene og identifisere hvor mange av dem er korrekte.
    - Relasjonsnøyaktigheten beregnes ved å telle antallet korrekte relasjoner og dele dette på det totale antallet mulige relasjoner.
    - Dette gir innsikt i hvor god modellen klarer å forutsi riktige relasjoner mellom ulike lokasjoner i datasettet.

4. Beregning av total nøyaktighet

    - Til slutt kombinerer vi lokasjonsnøyaktigheten og relasjonsnøyaktigheten for å beregne den totale nøyaktigheten
    - Den totale nøyaktigheten er gjennomsnittet av lokasjons- og relasjonsnøyaktighetene, og gir en helhetlig vurdering av hvor godt modellen klarer å forutsi riktige verdier i datasettet.

Fordeler

Denne algoritmen er en omfattende vurdering av nøyaktigheten til prediksjoner i datasettet ved å ta hensyn til både lokasjoner og relasjoner.

- Ved å innkludere både lokasjons- og relasjonsinformasjon i dybde, gir denne tilnærmingen et mer nyansert bilde av nøyaktigheten til prediksjonene.

Konseptuell matematisk formel av brukt algoritme

Let:

C_{loc} nummer av korrekte lokasjoner,

T_{loc} den totale mengden av lokasjoner.

C_{rel} nummer av korrekte relasjoner, tillater delvis rette.

T_{rel} den totale mengden av relasjoner.

Lokasjonsnøyaktigheten A_{loc} kan uttrykkes som forholdet mellom antall riktige forutsagte lokasjoner og det totale antallet forutsagte eller faktiske lokasjoner:
$A_{\text{loc}} = \frac{C_{\text{loc}}}{T_{\text{loc}}}$

Relasjonsnøyaktigheten A_{rel} kan uttrykkes som forholdet mellom antall riktige forutsagte relasjoner og det totale antallet forutsagte eller faktiske relasjoner:
$A_{\text{rel}} = \frac{C_{\text{rel}}}{T_{\text{rel}}}$

Den totale nøyaktigheten A_{total} kan uttrykkes som gjennomsnittet av lokasjons- og relasjons-nøyaktigheten:

$\text{A}_{\text{total}} = \frac{A_{\text{loc}} + A_{\text{rel}}}{2}$

6.4 GitHub

Om programkoden for webapplikasjon skulle være ønskelig å utforske er den tilgjengelig på GitHub fra denne lenken:

https://github.com/rakvarv/rkv009-masters

Kapittel 7

Resultat

7.1 Uthenting av geospatielle data

Å hente ut data ved hjelp av kunstig intelligens kan være en omfattende og utfordrende oppgave. Dette skyldes delvis de omfattende nyansene og tvetydigheten i språket. En annen grunn til vanskelighetene er AI's nåværende begrensninger. AI blir stadig mer avansert og bedre på flere oppgaver, men i sin nåværende tilstand er det fortsatt en betydelig mengde begrensninger som bør og må tas hensyn til. Etter omfattende testing ved bruk av ulike modeller tilgjengelig fra OpenAI, varierte resultatene ganske mye. Mange ulike tester er gjennomført, fra bruk av zero-shot læring, helt til en svært omfattende few-shot læring. Disse testene gir innsikt i både styrkene og svakhetene ved de ulike tilnærmingene, og gir et rikere perspektiv på hva som kreves for å oppnå pålitelige og nøyaktige resultater i dataekstraksjon ved hjelp av AI.

7.1.1 Zero-Shot læring

Når zero-shot læring blir brukt, sender ikke API-et et eneste eksempel på hva brukeren vil at AI-en skal gjøre eller hvordan resultatet bør se ut når API-et sender det tilbake til brukeren. Dette gapet i forståelsen blir tydelig når denne metoden tas i bruk. La oss ta et konkret eksempel: Jeg forteller API-et at oppgaven er å hente ut geospatiale data og returnere det i JSON-format. I denne første zero-shot-læringsprøven velger jeg å ikke beskrive en ontologi for å se hva API-et kan oppnå av seg selv. Når API-et sender tilbake resultatet, forsøker det å følge JSON-formatet slik det ville blitt tolket i Python. Men det er ikke alltid tilfellet. Uten spesifikke eksempler ender API-et ofte opp med å levere data i et format som ikke kan tolkes

som JSON av en maskin. Dette kan være så enkelt som bruk av enkle anførselstegn i stedet for doble, noe som kan forårsake feil i det påfølgende programmet fordi lokasjoner og relasjoner i JSON-objektet ikke kan parses riktig.

Når API-et til slutt lykkes med å sende tilbake et korrekt JSON-objekt, er det ikke alltid konsistent. For eksempel kan en nøkkel i JSON-objektet hete ”relationships”, men API-et returnerer den kanskje som ”relations”. Dette gjør datahåndteringen mer utfordrende og uforutsigbar.

Når vi dykker ned i ekstraksjonsprosessen av disse dataene, blir begrensningene til GPT i forhold til språk tydelige. I denne forskningen bruker jeg nyhetstekster fra Vestland som kilde. Siden GPT er trent på et spesifikt korpus av tekstdata, kan den ha vanskeligheter med å skille mellom hva som er en lokasjon og hva som ikke er det. For eksempel kan et jordskjelv bli oppfattet som en navngitt lokasjon, selv om det egentlig er et midlertidig fenomen som ikke har en fast geografisk tilhørighet. Dette skyldes ikke bare begrensningene til GPT, men også mangelen på spesifisitet i læringen av geografiske entiteter som skal anerkjennes som lokasjoner.

Her kommer One-Shot og Few-Shot læring inn i bildet, hvor vi kan tilpasse modellen bedre til å gjenkjenne og forstå slike subtile forskjeller i dataene. Det er gjennom å eksperimentere med ulike tilnærminger og treningsmetoder at vi kan forbedre AI-ens evne til å trekke ut nøyaktig og meningsfull informasjon fra komplekse ustrukturerte tekster.

Beregning av nøyaktighet for Zero-Shot læringen ble ikke fullført på grunn av det store mangelen av et konsistent JSON format fra API’et, som gjør utføring av denne evalueringen vesentlig tidskrevende og ikke verdt resultatet det hadde gitt med tanke på hvor mye tid jeg har til å utføre det.

7.1.2 One-Shot Learning Approach

Når one-shot læring blir tatt i bruk, ligner det i stor grad på zero-shot metoden, men med én vesentlig forskjell: API-et mottar ett eksempel på hva brukeren forventer at AI-en skal utføre eller hvordan resultatet skal se ut. I denne tilnærmingen til one-shot læring beskriver jeg enkelt hvordan ontologien fungerer og inkluderer noen av relasjonstypene som er relevante. Jeg

sender også inn et eksempel på forventet output, vanligvis i form av et JSON-objekt som har samme struktur som ønsket resultat. Dette eksempelet bidrar til at API-et forstår hvilket format det skal returnere, spesielt hvis det oppdager en lokasjon nevnt i nyhetstekstene som blir behandlet.

Takket være dette eksempelet kan API-et sende tilbake data i riktig format som kan tolkes som JSON og brukes umiddelbart. Når det gjelder ekstraksjon av geospatiale data, har denne metoden sine treff og bommer. Noen ganger, når API-et ikke identifiserer noen form for lokasjon i nyhetsteksten, antar det at det ikke er nødvendig å returnere JSON-objekter, siden det ikke finnes noen identifiserbare lokasjoner. Dette betyr at modellens nøyaktighet når det gjelder datautvinning ikke er ekstremt høy, men heller ikke relativt dårlig. Det er et balansespill mellom å oppnå presise resultater og å unngå unødvendig kompliserte og feilaktige utdata. Det viser at selv med enkle eksempler og veiledning kan AI-en fortsatt ha utfordringer med å tolke komplekse situasjoner nøyaktig.

Både for GPT-3, og GPT-4 så presterer de veldig bra på å ekstrahere alle lokasjonene i teksten. De aller fleste tifellene vil legge på en plass mellom 40-50% for en total nøyaktighet. Gjennomsnittlig så får GPT-4 modellen en total nøyaktighet på 47%, mens GPT-3 ligger på 41%. Lokasjons nøyaktigheten for GPT4 ligger på 68.89%, mens i GPT3 ligger den på 50.90%. Den svakeste delen av nøyaktigheten kommer ned til beregningen av relasjonsnøyaktigheten. Relasjonsnøyaktigheten for GPT3 ligger på 31.20%, mens GPT4 ligger på så lite som 27%.

7.1.3 Few-Shot læring med finjustert GPT modell

Finjustering av GPT modell

For å evaluere modellens presisjon på det høyeste nivået, kan vi benytte oss av en teknikk som kalles finjustering. Dette innebærer i hovedsak å tilføre data til GPT-modellen som man ønsker å forbedre, og deretter trene modellen i henhold til dens formål og metode. Selv om prosessen kan virke kompleks ved første øyekast, er den egentlig ganske grei. Dataen som mates inn i GPT-modellen er i praksis en samling av samtalehistorikk i form av JSONL-format. Når du leverer denne dataen til API-et, er det som om du allerede har gjennomført disse samtalene med GPT, og dermed vil modellen huske hva den har svart tidligere i forhold til hvilke meldinger

du sender inn, noe som Bergmann (2024) skriver en del om med finjustering av maskinlæringsmodeller.

Hensikten med dette er å forbedre modellens nøyaktighet på best mulig måte. Ifølge OpenAI (2024) kreves det minst 10 eksempler på samtalehistorikk for å finjustere en modell. Ettersom jeg har tilgang til 140 nyhetstekster, velger jeg å kun benytte 10 av disse. Dette gjøres for å unngå potensielle problemer med overtilpasning. Overtilpasning oppstår når modellen trenes for grundig på tilgjengelige data, slik at den kun gjengir svarene den ble matet under treningsfasen, som da anses som riktige svar.

Dette kan gi et forvrengt bilde av modellens nøyaktighet på grunn av den begrensede mengden data som ble brukt under treningen. Det er derfor viktig å være bevisst på valg av treningsdata for å sikre en mer pålitelig vurdering av modellens ytelse.

Strukturen av ontologien jeg har utviklet er også innkludert i denne finjusteringen. Her forklarer jeg alle de forskjellige typer lokasjoner jeg har innkludert, i tillegg til alle de forskjellige relasjonene som hører til enkelte typer lokasjoner. Dette er system meldingen jeg skrev til finjusteringen av modellen:

> You are trained to extract named graphical locations and the geographical relationships between them from news-texts. And only the locations from the news-text provided should be included in the output. The only locations that will be picked out are permanent named, and locally/globally recognized as a permanent location in that geographical area. Once a news-text/text is fed to you, you need to extract the relationships and locations in a specific format, which is JSON. The JSON format looks like this {'locations': [{'name': 'locationName', 'type': 'Location', 'entity': 'locationType'}], 'relationships': ['from': 'fromLocation', 'relation': 'relationBetweenLocations', 'target': 'targetLocation']}. You must keep this format, and the JSON keys provided must have the same keyname as these ones just provided. Different locations have different types, and different types of locations have different relationships it can inherit. An example is that a body of water that is bordering a landmass is not a part of that landmass, but just bordering it, unless the land-

mass encapsulates it. You must follow a specific ontology, which describes these. This is the ontology. The different types of locations you have available in the entity key is: [Land, Ocean, Road, Mountain]. Land is categorized as permanent named geographical locations on land. Land can include forests, bodies of water, cities, and so on. Examples of locations of this entity is: [Vestland, Bergen, Åsane, Vik, Loddefjord]. The relationships belonging to Land is: [isNearby, contains, isAdjecentTo]. Roads is another entity type, examples of roads are: [highways, freeways, motorways, roads]. A road can not be a mountain trail in this instance. All relations linked to a road is: [isNearby, isPassingThrough, isAdjecentTo] Third entity class, which is Ocean. Ocean is a named large body of water. An ocean can also encapsulate islands. Relations belonging to oceans are: [contains, isNearby, isAdjecentTo]. Last location entity is Mountain. Mountains are recognized as tall landmass tops, which are named. Examples of mountains are: [Filefjell, Hemsedalsfjellet, Lyderhorn, Ulriken]. Relations linked to mountains are: [isNearby, overlooks]. All location entity types can have the relationships: [isNearby, isAdjecentTo]. All location specific relations are relations that can only be used from that location entity type, which is really important to follow. Relationship keys must be [from, relation, target]. Make sure every JSON key and value has quoutes surrounding it, and is not left empty.

Nøyaktighet med finjustert modell

Når vi tar i bruk Few-Shot læring, integrerer jeg eksempler som illustrerer hvordan det ønskede returnerte svaret skal se ut, samt hvordan modellen skal utføre de oppgavene den er instruert til å utføre. Denne tilnærmingen til Few-Shot læring involverer analysen av 10 nyhetstekster som inneholder informasjon om lokasjoner, relasjoner eller deres fravær. Deretter gir jeg modellen det korrekte svaret i henhold til min gullstandard for dette eksperimentet. Disse eksemplene danner grunnlaget for modellens forståelse av hvordan den skal ekstrahere lokasjonene og relasjonene. Dette inkluderer en detaljert analyse av ontologien som ble utviklet for eksperimentet, som beskriver hvilke typer lokasjoner og relasjoner som er inkludert, samt hvordan de forholder seg til hverandre. Denne grundige tilnærmingen sikrer at modellen har en omfattende forståelse av konteksten den opererer i, og øker nøyaktigheten og påliteligheten til dens

resultater.

Few-Shot-eksemplet innebærer også finjustering av GPT-3-Turbo-modellen. Mens GPT-4-Turbo-modellen ikke ble finjustert og derfor ikke ble testet for dette formålet, gjennomgikk den imidlertid Few-Shot-læring. Resultatene av finjusteringen av GPT-3-Turbo var overraskende bedre sammenlignet med tidligere iterasjoner av GPT-3-modellen. Uten finjustering oppnådde GPT-3 en nøyaktighet på 41%. Etter finjusteringen økte nøyaktigheten til den modifiserte modellen til imponerende 51.9%. Dette representerer en bemerkelsesverdig forbedring på hele 10.9% sammenlignet med den tidligere versjonen. Relasjonene økte fra beskjedne 31.2% til beskjedne 34.42%. Lokasjonsnøyaktigheten for modellen økte også betydelig fra 50.9% til 69.5%, dette demonstrerer den betydelige effekten av finjusteringen på modellens ytelse for lokasjonene, selv om relasjonene hadde en veldig liten betydelig endring. GPT-4-Turbo sin ytelse, selv uten finjustering gjorde et veldig bra hopp fra vanlig GPT-4 med 47% til GPT-4-Turbo med 78% total nøyaktighet. Her ligger lokasjonsnøyaktighet på et imponerende 76.26% nøyaktighet. Her imponerte også relasjonsnøyaktigheten med 79.96% nøyaktighet fra så lite som 27%. Disse resultatene understreker potensialet og effektiviteten av Few-Shot-læring i å forbedre ytelsen til naturlig språkbehandlingsteknologier som GPT.

7.2 Ontologi

Ontologien som er utviklet for å ekstrahere den geospatiale dataen og relasjonene er nøye konstruert for å kunne skille mellom ulike typer lokasjoner. Den tar hensyn til en rekke forskjellige geografiske elementer, inkludert landområder, veier, fjell og hav. Hver av disse lokasjonstypene har sine egne distinkte egenskaper og relasjoner som bidrar til å identifisere dem og deres samspill med andre geografiske områder. For eksempel kan en vei ha relasjoner som indikerer dens tilknytning til andre veier, byer den passerer gjennom, eller landskap den krysser. På samme måte kan et fjell ha relasjoner som angir dets høyde, nærliggende fjelltopper, eller eventuelle stier eller turstier som fører til det. Havområder kan også ha spesifikke relasjoner som indikerer dybde, strømmer eller tilknytning til kystområder og øyer. Denne detaljerte ontologien muliggjør presis identifisering og forståelse av ulike geografiske enheter, og legger dermed grunnlaget for nøyaktig og informativ datautvinning fra nyhetskilder.

7.2.1 Lokasjonstyper

Landområdet Ontologien omfatter fire distinkte typer lokasjoner, som inkluderer landområder, veier, hav og fjell. Den første lokasjonstypen, landområder, er karakterisert ved å være permanente geografiske områder på land. Disse områdene kan strekke seg over en spesifikk lengde og inkluderer byer, bydeler, skoger, parker og til og med hele land. Landområder kan inkludere vann, men de kan ikke omfatte havområder. Eksempler på landområder i Vestland, Norge, inkluderer byer som Bergen, Åsane, Loddefjord og Vik. Listen over navngitte landområder kan også inneholde mindre geografiske enheter som skoger, parker og andre kjente områder. Det er viktig at disse navngitte stedene er anerkjent som offisielle lokasjoner på landsbasis for å bli inkludert i ontologien.

Veier Veier er en annen viktig lokasjonstype i ontologien. Disse inkluderer motorveier, hovedveier, gater og alle andre former for veier som binder sammen forskjellige geografiske områder. Veier har spesifikke egenskaper og relasjoner som bidrar til å identifisere dem, for eksempel deres navn, lengde, tilknytning til andre veier og tilknytning til nærliggende byer eller landemerker.

Havområder Havområder representerer et annet viktig element i ontologien. Disse områdene omfatter alle former for store vannmasser som strekker seg over store deler av jordens overflate. Havområder kan ha forskjellige egenskaper og relasjoner, inkludert dybde, strømmer, havstrømmer og tilknytning til kystområder og øyer.

Fjell Fjell utgjør den fjerde lokasjonstypen i ontologien. Disse områdene kjennetegnes av sine høye topper og ruvende landskap. Fjell kan ha spesifikke egenskaper som høyde, tilknytning til andre fjellkjeder eller fjelltopper, samt tilstedeværelsen av stier eller turstier.

Samlet sett muliggjør denne omfattende ontologien nøyaktig identifisering og kategorisering av ulike geografiske enheter, og gir dermed et solid grunnlag for effektiv og pålitelig datautvinning fra tekstbaserte nyhetskilder.

7.2.2 Relasjonstyper

Som tidligere forklart i denne seksjonen, har de ulike typene lokasjoner ulike relasjoner som identifiserer deres forbindelser med andre lokasjoner. Disse relasjonene opererer på en en-til-mange basis, noe som betyr at alle relasjonene som tilhører for eksempel landområder, er de forbindelsene som landområder har til andre typer lokasjoner, og ikke omvendt. Det eneste unntaket er relasjonene som er felles for alle lokasjonstypene. Her er de forskjellige relasjonstypene.

Land Landområder har svært enkle relasjoner og klare grenser med hensyn til deres utstrekning; de trenger ikke å trenge seg inn på andre områder. Derfor har de bare relasjoner med sin egen lokasjon i forhold til andre basert på grenser. De relasjonene landområder har, inkluderer: ”isNearby”, ”contains”, og ”isAdjacentTo”. ”isNearby-relasjonen” betyr at lokasjonen som er nevnt, er i nærheten av en annen geografisk entitet, men deler ingen grense med denne entiteten. Disse relasjonene har ulike nivåer av omfang for å forenkle forståelsen og varierer i henhold til det totale området som alle lokasjonene er tilknyttet. Relasjonen "containsrefererer til at den nevnte lokasjonen omfatter en annen geografisk entitet innenfor sine fysiske grenser. Et eksempel på dette er at landet Norge ”contains” byen Bergen. ”isAdjacentTo” betyr at den nevnte lokasjonen er i nærheten av en geografisk entitet og deler en grense med denne entiteten.

Veier Relasjonstypene som tilhører veier ligner veldig på de som tilhører landområder, bortsett fra én forskjell. De relasjonstypene er: ”isNearby”, ”isPassingThrough”, og ”isAdjacentTo”. ”isPassingThrough” refererer til at veien som er nevnt, går gjennom en bestemt geografisk entitet. Et eksempel er at vei1 ”isPassingThrough” by1. ”isAdjacentTo” i denne sammenhengen betyr at veien som er nevnt, følger langs en geografisk entitet, for eksempel et vann eller en annen vei. ”isNearby-relasjonen” betyr det samme som for landområder, og brukes bare til å beskrive at veien ligger i nærheten av en geografisk entitet, men ikke deler grense med den.

Fjell Fjell har færre relasjoner i forhold til andre typer lokasjoner. De relasjonene som tilhører fjell er: ”isAdjacentTo” og ”overlooks”. ”overlooks” betyr at det nevnte fjellet har utsikt

over visse andre geografiske entiteter fra en eller annen posisjon på fjellet. ”isAdjacentTo” har samme betydning som for andre lokasjoner, og indikerer at fjellet er i nærheten av en annen geografisk entitet og deler en grense med den. Et eksempel på dette er at et fjell kan være "adjacenttil en innsjø.

Hav Relasjonene knyttet til havet er: ”isNearby”, ”contains”, og ”isAdjacentTo”. I dette tilfellet betyr ”isNearby” at et hav ligger langs grenseområdet til et land. For eksempel, Atlanterhavet er ikke en del av Storbritannia, men det grenser til Storbritannia. ”Contains” refererer til at havet kan inneholde geografiske entiteter som øyer. "IsAdjacentTobetyr at havet er i nærheten av et annet geografisk område, selv om de ikke deler en direkte grense.

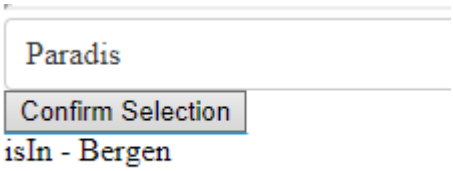
RELASJONER	contains	isNearby	isAdjecentTo	overlooks	isPassingThrough
LAND	X	X	X		
VEIER		X	X		X
FJELL	X	X		X	
HAV	X	X	X		

Tabell 7.1: Tabell som inneholder alle lokasjonstyper med deres rette mulige relasjoner

7.3 Geospatial relasjons uthenting/visualisering

7.3.1 Relasjons uthenting

Måten relasjonene er hentet ut fra dataene er ved bruk av SPARQL. Mer om SPARQL og lignende språk kan du lese mer om i seksjon 6.2.4. Relasjonsuthenting er implementert i brukergrensesnittet brukeren har tilgang til. Dette er i form av en dropdown meny hvor du kan velge en lokasjon, trykke en annen knapp, og få ut alle relasjoner som går utifra den valgte lokasjonen. Dette er den del av det som bidrar til automasjons journalistikk, med at en journalist kan velge lokasjoner, og se alle plasser som er relatert til den plassen, og kan potensielt bidra til nyhetslaging.



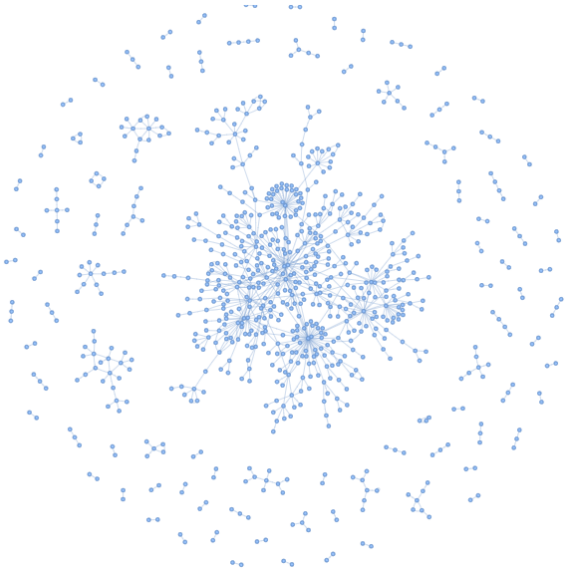
Figur 7.1: Eksempel på faktisk relasjonsuthenting fra brukergrensesnitt

```
def get_relationships_for_location(location, ttl_content):
    g = rdflib.Graph()
    g.parse(data=ttl_content, format="turtle")

    # Adjusted SPARQL query to get both relationship type and target location
    query = f"""
    PREFIX loc: <http://example.org/locations/>
    PREFIX rel: <http://example.org/relationships/>

    SELECT ?relationshipType ?otherLocation WHERE {{
        loc:{location} ?relationship ?otherLocation .
        BIND(STRAFTER(STR(?relationship), "http://example.org/relationships/") AS ?relationshipType)
    }}
    """
```

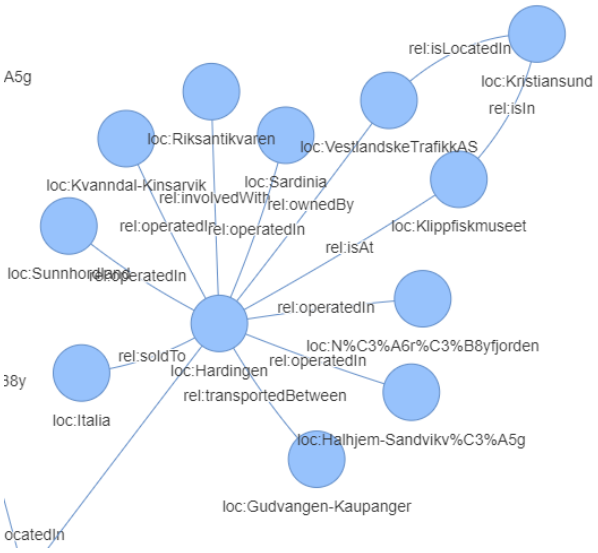
Figur 7.2: RDF kode for uthenting av relasjoner fra lokasjoner

7.3.2 Geospatial relasjons visualisering

Relasjons visualiseringen er i form av en kunnskapsgraf tilgjengelig i brukergrensesnittet. Kunnskapsgrafen i seg selv kan justeres nærmere til brukeren, man kan flytte på eventuelle noder etter brukerens vilje. Man kan markere alle relaterte noder til en spesifkk node i en mørk farge for å visualisere alle relasjonene til en spesifikk geografisk entitet.



Figur 7.3: Eksempel på faktisk relasjons visualisering fra brukergrensesnitt



Figur 7.4: Eksempel på faktisk relasjons eksempel justert nærmere

Kapittel 8

Diskusjon

8.1 Måling av geospatial uttrekk med GPT

8.1.1 Effektiviteten med x-shot læring

zero-shot På grunn av at GPT er trent på et spesifikt korpus av tekst, håndterer den teksten på en måte som samsvarer med sin treningsdata. Den tolker derfor den innsendte ledeteksten på en annen måte enn det du kanskje ønsker. Dette fenomenet illustreres tydelig når jeg sender inn min ledetekst. Vanligvis har jeg en systemtekst som beskriver GPT som en kunstig intelligensmodell spesialisert i å ekstrahere lokasjons- og relasjonsdata fra nyhetstekster, og deretter returnere svaret i form av en JSON-struktur, som blir servert sammen med nyhetsteksten. Denne tilnærmingen gir GPT en viss grad av kreativ frihet til å formidle informasjonen på sin egen måte, i stedet for å følge et forhåndsdefinert JSON-format. Dessverre fører dette til at GPT tolker hva som skal regnes som en "lokasjon", og hvilke typer "relasjoner" den bør inkludere i svaret, på en måte som ikke alltid samsvarer med våre forventninger. Vi hadde håpet på konkrete lokasjoner, spesifikke relasjoner og en fastsatt JSON-struktur. Et eksempel på feil JSON format som modellene ofte kom med var hva nøklene til relasjons delen av objektet skulle navngis. Noen eksempler er at den kom med "source" istedenfor "from", som er noe som skapte problemer i håndteringen av evaluering av nøyaktigheten av modellene. Nøyaktighet ble ikke målt for zero-shot på grunn av mangelen av en konsistent output, som da viser at zero-shot gjerne ikke gir det aller beste resultatet i forhold til et konsistent og korrekt svar i forhold til definert gullstandard. Noen eksempel på hvordan lokasjoner og relasjoner gjerne så ut fra zero learning kan se slik ut:

API svar:

- loc:PRIVATADRESSE rel:soldFromTo loc:PersonO
- loc:E10Presteneset rel:isIn loc:Nordland
- loc:Enebolig rel:isLocatedIn loc:Fantoft
- loc:GulatingLagmannsrett rel:ruledInFavorOf loc:St1Norge

System melding: I am an AI model and I have been trained to extract and structure information about locations and their geo spatial relationships from texts and all locations bordering the mentioned locations. The output will be in a valid JSON format using double quotes, not single. Do not specify you are writing the output, only write the JSON. If the locations include whitespaces, make the location in camel case writing. Here are a few examples

Dette er et eksempel på hvordan lokasjonene og relasjonene gjerne kommer ut ved hjelp av et zero-shot eksempel. Disse eksemplene er hentet fra en TTL fil som inneholder alle RDF'ene fra zero-shot læringen. Eksemplene vist gir oss en idé om hvordan GPT med mye kreativ frihet kan skape flertydige svar, som i samme type lokasjoner og relasjoner, men forskjellige navn for samme relasjonene. Ene eksemplet er at "PRIVATADRESSE SolfFromTo Person0". System meldingen sier at det er loasjoner og geo spatielle forhold som skal komme med. Her at vi ikke en navngitt lokasjon, men en direkte addresse til en privatperson, hvor relasjonene beskriver at dette huset ble solgt fra en person til en annen, noe som er en relasjon vi ikke er interessert i.

Et annet eksempel er hvor den kombinerer to forskjellige navngitte geografiske entiteter. I dette tilfelle kombinerer den veien E10 og landområde Presteneset. Denne hender gjerne i tekster hvor stedet blir nevnt rett etter veien. Da gir det mer mening at E10 og presteneset er to forksjellige lokasjoner med en relasjon som binder de. Deretter kan du ha relasjonen at begge er relatert til Nordland, men hver for seg.

Nå kommer vi der til problemet med flertydige svar. Det eksempelet som nettopp ble nevnt med E10Presteneset og Nordland, så har vi en relasjon som heter "isIn". Dette er den rette releasjonstypen vi ser etter, men siden denne relasjonen aldri ble nevnt til API'et, så er det

bare API'et som har logisk kommet frem til dette navnet i forhold til teksten. Men problemet er at selv om GPT legger merke til samme type relasjoner for andre lokasjoner, så velger den ikke relasjonen veldig konsistent. For eksempel med Enebolig og Fantoft, så istedenfor å bruke 'isIn' spm den tidligere hadde brukt, så bruker den "isLocatedIn", selv om de betyr det samme. Dette kan skape problemer i datahåndtering etterpå, på grunn av API'ets mangel av et konsistent svar.

Siste eksempelet bruker Gulating Langmannsrett, som er tinghuset i Bergen. Dette er teknisk sett også en lokasjon, men problemet er at dette er ikke en permanent geografisk og navngitt lokasjon. I tillegg så går den ikke under ontologien spesifisert tidligere i kapittel 7.2. Her bruker de også resultatet fra en sak i langmansretten som relasjonen, hvor den heter "ruledInFavorOf", hvor lokasjonen som blir relatert til, er "St1Norge". Disse eksemplene viser hva GPT får til viss du gir den nok kreativ frihet og ingen eksempler på¨hvordan sluttresultater skal se ut. Da får du problemer med syntax, lokasjonstyper, og relasjonstyper.

one-shot Når det gjelder zero-shot-læring, er one-shot-læringen et skritt videre hvor vi presenterer modellen med et enkelt eksempel på hvordan resultatet av ledeteksten skal se ut. Dette er der JSON-formatet blir avgjørende. I systemmeldingen som sendes til modellen for å be om en respons, kan vi beskrive et JSON-format. Men på grunn av mangelen på eksempler i zero-shot-læringen har modellen ingen praktiske retningslinjer for hvordan dette skal se ut. I one-shot-læringen gir vi derimot et konkret eksempel på hvordan responsen bør se ut i forhold til en nyhetstekst som presenteres for modellen. Generelt sett fungerer one-shot-strukturen ganske bra. Imidlertid oppstår utfordringer når det gjelder faktiske relasjoner og lokasjoner. Vi gir ikke tilstrekkelig med eksempler på hvilke typer lokasjoner som bør inkluderes, eller hvordan vi skal kategorisere dem. Dette resulterer i lavere nøyaktighet for modellen i stor grad.

Fra tid til annen kan modellen produsere feilaktige JSON-objekter, men dette forekommer langt sjeldnere enn i zero-shot-læringen. Dette illustreres tydelig når vi sammenligner nøyaktigheten av one-shot-læringen med zero-shot-læringen. GPT-3-modellen oppnådde en total nøyaktighet på 31.20%, mens GPT-4 ligger på en enda lavere 27%. Dette viser hvor stor innflytelse et enkelt eksempel har på GPT, og hvordan det kan redusere modellens troverdighet når den skal produsere nøyaktige verdier som potensielt vil bli brukt av journalister. Noen eksempel på hvordan lokasjoner og relasjoner gjerne så ut fra zero learning kan se slik ut:

API svar:

- loc:Haugesund rel:isIn loc:NordRogaland
- loc:HordalandFolkeblad rel:isLocatedIn loc:Hordaland
- loc:InjuredDriver rel:transportedTo loc:HospitalInF%C3%B8rde .
- loc:Liavannet rel:isIn loc:Bergensomr%C3%A5det

System melding: I am an AI model and I have been trained to extract and structure information about locations and their geo spatial relationships from texts and all locations bordering the mentioned locations. The output will be in a valid JSON format using double quotes, not single. Do not specify you are writing the output, only write the JSON. If the locations include whitespaces, make the location in camel case writing. Here are a few examples

Resultatet fra one-shot og zero-shot er veldig lik, siden eneste forskjellen er at du gir et eneste eksempel på hvordan svaret fra AOI'et skal se ut. Da må du ha et veldig bra eksempel som dekker veldig mange forskjellige problemer. Den store forskjellen mellom zero-shot og one-shot er at den klarer å ha et konsistent JSON-format. Istedenfor at API'et gir et JSON-format som den tror er det du ser etter. så i eksempelet som blir servert til API'et så gir du et eksempel på nøyaktig hvordan JSON-formatet skal se ut, og hva navnet på nøklene skal være. Selv om formatet på svaret er som oftest rett, så har den de samme probleme som zero-shot læringen har, bare i litt mindre grad.

Et eksempel som ofte kommer opp selv i one-shot læring er å bruke organisasjoner eller regjeringen som lokasjoner. Dette vises i eksempelet hvor det står "HordalandFolkeblad isLocatedIn Hordaland". Dette viser at den tar Hordaland Folkeblad som en lokasjon, istedenfor en organisasjon. Dette skjer stadig ofte i one-shot lærings-svarene.

few-shot I few-shot-læring er det hvor vi virkelig spesifiserer hvordan svaret fra ledeteksten skal se ut, og hvordan den skal håndtere lokasjonene og relasjonene i teksten. Det som er spesielt med few-shot-læringen er at det ikke bare handler om å inkludere 3-4 eksempler. Her kan det være en mengde eksempler over en bred skala. Det gir oss muligheten til å gi så mange eksempler som mulig, men det må være en balanse for å unngå overfitting i modellen. IBM (2024b) forklarer mer om overfitting.

Som forklart i resultatkapittelet, inkluderte jeg 10 eksempler, som utgjør 6.94% av den totale mengden nyhetstekster som ble inkludert i datasettet som ble servert til GPT. Dette antallet eksempler er 10 ganger så mange som det som ble brukt i one-shot-læringen, og dette viser seg tydelig i resultatene. Overraskende nok viser både GPT-3 og GPT-4-Turbo ekstremt forskjellige resultater når det gjelder korrekte relasjoner. For relasjoner oppnådde GPT-3 en nøyaktighet på 34.4%, mens GPT-4 oppnådde en nøyaktighet på 78%. Det bemerkelsesverdige er at selv om et svar fra GPT-4 inneholdt et feilaktig JSON-objekt, så hadde GPT-4 en vesentlig høyere nøyaktighet for relasjonene, noe som er svært overraskende med den enorme forskjellen.

Dette kan også skyldes at few-shot-læringen til GPT-4 inkluderer de samme nyhetstekstene og svarene som GPT-3-modellen ble finjustert med, noe som betyr at GPT-3 er mer trent på disse svarene, men den nyere teknologien og teknikkene til GPT-4 viser hvor bedre den er til denne oppgaven enn GPT-3. Nøyaktigheten når det gjelder lokasjoner for begge modellene er også overraskende like. GPT-3 oppnådde en nøyaktighet på 69.52%, mens GPT-4 oppnådde en nøyaktighet på 76.26%. Selv om gapet mellom dem er noe større her, er det fortsatt bare 6% fra hverandre. Dette viser at treningen på gullstandarden for lokasjoner har blitt nøyaktig innlemmet av begge modellene, og derfor har økt nøyaktigheten betydelig. Men det viser også hvor nær toppen av ytelsen GPT-3 og GPT-4 er når det gjelder lokasjonsekstraksjon.

På den totale nøyaktigheten oppnådde GPT-3 en nøyaktighet på 51.97%, mens GPT-4 oppnådde en nøyaktighet på 78%. Det var forventet at GPT-4 ville utføre oppgavene bedre enn GPT-3 totalt sett, men det var overraskende å se at på et one-shot-læringsnivå utførte GPT-3 oppgavene bedre. Dette kan skyldes flere grunner, blant annet at GPT-4 har tilgang til et større tekstkorpus, som potensielt kan ha komplisert dataene og resultert i feilaktige svar.

API svar:

- loc:NorgesHandelhøyskole rel:isIn loc:Bergen
- loc:Storeklubben rel:isIn loc:Askøy
- loc:SognOgFjordane rel:contains loc:Gudvangtunnelen

- loc:Borgundstunnelen rel:isNearby loc:Steinklepp

System Melding: You are trained to extract named graphical locations and the geographical relationships between them from news-texts. And only the locations from the news-text provided should be included in the output. The only locations that will be picked out are permanent named, and locally/globally recognized as a permanent location in that geographical area. Once a news-text/text is fed to you, you need to extract the relationships and locations in a specific format, which is JSON. The JSON format looks like this {'locations': [{'name': 'locationName', 'type': 'Location', 'entity': 'locationType'}], 'relationships': ['from': 'fromLocation', 'relation': 'relationBetweenLocations', 'target': 'targetLocation']}. You must keep this format, and the JSON keys provided must have the same keyname as these ones just provided. Different locations have different types, and different types of locations have different relationships it can inherit. An example is that a body of water that is bordering a landmass is not a part of that landmass, but just bordering it, unless the landmass encapsulates it. You must follow a specific ontology, which describes these. This is the ontology. The different types of locations you have available in the entity key is: [Land, Ocean, Road, Mountain]. Land is categorized as permanent named geographical locations on land. Land can include forests, bodies of water, cities, and so on. Examples of locations of this entity is: [Vestland, Bergen, Åsane, Vik, Loddefjord]. The relationships belonging to Land is: [isNearby, contains, isAdjecentTo]. Roads is another entity type, examples of roads are: [highways, freeways, motorways, roads]. A road can not be a mountain trail in this instance. All relations linked to a road is: [isNearby, isPassingThrough, isAdjecentTo] Third entity class, which is Ocean. Ocean is a named large body of water. An ocean can also encapsulate islands. Relations belonging to oceans are: [contains, isNearby, isAdjecentTo]. Last location entity is Mountain. Mountains are recognized as tall landmass tops, which are named. Examples of mountains are: [Filefjell, Hemsedalsfjellet, Lyderhorn, Ulriken]. Relations linked to mountains are: [isNearby, overlooks]. All location entity types can have the relationships: [isNearby, isAdjecentTo]. All

> location specific relations are relations that can only be used from that location entity type, which is really important to follow. Relationship keys must be [from, relation, target]. Make sure every JSON key and value has quoutes surrounding it, and is not left empty.

Selv med few-shot læring så hender det at potensielle feil kommer med i API svaret. Her i few-shot læringen er det gitt 10 nyhetstekster med korrekt API svar. Svarene som kommer fra few-shot læringen handler mindre om syntaks feil eller uthenting av lokasjoner. few-shot sine feil handler mer om hvilke bruk av relasjon i forhold til ontologi er tatt i bruk. Her har vi noen eksmpler. Første relasjon bruker Norges Handelhøyskole som en lokasjon, dette er i mitt tilfelle en feil lokasjonstype. Dette er noe som også er ganske vanlig gjennom zero-shot og one-shot, men er vesentlig sjeldnere i few-shot, noe som vi ser i forhold til resultatene. Dette er tilfelle for både GPT3 og GPT4, der few-shot har størst problemer er ved følging av ontologi i relasjonsuthentingen. I første eksempelet med Handelshøyskolen så har vi bruk av relasjonen ”isIn”, som referer til at den lokasjonen er innen for en annen geografisk entitet, som i dette tilfelle er Bergen. Denne relasjonen i seg selv gir mening, men er ikke relasjonen som er spesifisert i treningen til å bli brukt som relasjon. Denne API’en burde ta i bruk relasjonen ”contains”, som er den som er spesifisert i treningen av API’et. Det kan være flere grunner til at dette hender, hvor det viktigste av de er at vi ikke har veldig mye data å trene modellen på, hvor vi samtidig unngår overfitting. Å bruke flere eksempler gir API’et et større grunnlag for hvilke relasjoner som skal brukes i hvilke tilfelle.

Et annet eksempel er relasjonen mellom ”Sogn og Fjordane”, og ”Gudvangtunnelen”. Dette er et annet veldig bra eksempel på bruken av spesifikke relasjoner i spesifikke tilfeller. Her beskriver relasjonen at Sogn og Fjordane inneholder en geografisk entitet som heter Gudvangtunnelen. I forhold til ontologien i system meldingen og treningen gjort på modellen, så når en tunnel er innenfor en spesifikk geografisk entitet, så skal modellen bruke relasjonen ”isPassingThrough”. Hva denne relasjonen betyr er også beskrivet i treningen til modellen. Hadde mål lokasjonen ikke vært en tunnell, hadde denne relasjonene vært helt korrekt i forhold til ontologien. Men her kommer vi også til problemet hvor vi har begrensede treningsdata, som leder til usikkerhet i modellen.

8.1.2 Begrensninger i GPT

OpenAI sin teknologi GPT har et veldig stort potensial for kompleks tekst håndtering, men den har noen limitasjoner i forhold til hvordan den tolker instruksene og tilgjengelig data. Som tidligere forklart så hallusinerer GPT i form av at den har bare tilgjengeligheten til et spesifikt korpus av tekster, og ingenting mer. På grunn av dette, kommer den over noe som den faktisk ikke vet, eller er kraftig utdatert, så gir GPT deg et svar den tror er rett, selv om det egentlig ikke er det. Dette kan skape problemer, spesielt i forhold til lokasjoner og relasjoner. Lokasjoner og relasjoner kan være veldig komplekse temaer som krever veldig mye nøyaktig data.

Utviklet ontologi sammenlignet Med GPT's tekst-håndtering

GPT-modellene er trent på eksisterende data, og deres kunnskap strekker seg bare så langt som den tilgjengelige informasjonen tillater. Derfor kan det hende at det ikke er tilstrekkelig informasjon tilgjengelig om spesifikke typer ontologier. I mitt tilfelle benytter jeg en geospatial ontologi som jeg har utviklet for å effektivt håndtere ulike geografiske lokasjoner og deres komplekse relasjoner. Denne ontologien er grundig utforsket og beskrevet i kapittel 7.2, hvor dens betydning og anvendelighet blir grundig utforsket og diskutert. Ved å benytte denne ontologien, har jeg vært i stand til å oppnå dypere innsikt og forståelse innenfor feltet geospatial dataanalyse, og det har vært avgjørende for utforskningen av komplekse geografiske sammenhenger.

Da reiser spørsmålet seg: Er min ontologi bedre enn GPT-modellenes tekst håndtering? Selv om GPT-modellene gjerne er trent på hvordan den skal håndtere ulike ustruktuerte tekster, er det ikke alltid gitt at de er like konsistent eller nøyaktige, spesielt når det gjelder geografiske enheter og deres innbyrdes relasjoner. Dette ble tydelig da jeg testet modellens nøyaktighet og oppdaget at den ikke alltid fanger opp kompleksiteten i relasjonene mellom ulike entiteter. Et konkret eksempel på dette er når en geografisk enhet på land, som grenser til et vann, feilaktig tolkes som om vannet ligger innenfor den geografiske enheten på land. Imidlertid er dette ikke alltid tilfelle, da landet ikke nødvendigvis omkranser hele vannet, eller omvendt. I dette tilfellet kunne den tidligere påstanden ha vært korrekt, men på grunn av den spesifikke geografiske konteksten, er ikke denne relasjonen gyldig. Min ontologi tar høyde for dette og passer på at

spesifikke typer lokasjoner kan bare ha spesifikke typer relasjoner i forhold til andre typer lokasjoner.

Et annet viktig aspekt er at min ontologi tar utgangspunkt i navngitte geografiske lokasjoner, som er faste og varige. Dette skiller seg fra noen av GPT-modellene som kan strekke seg utover disse grensene, som for eksempel når den eventuelt inkluderer jordskjelv som en av de navngitte lokasjonene. Selv om GPT-modellenes tolkning ikke er direkte feil, da jordskjelvet faktisk har hendt på en geografisk lokasjon, så er jordskjelvet i seg selv ikke selve lokasjonen. Dermed blir det ikke en gyldig lokasjon i denne konteksten. Dette viser hvordan min ontologi, ved å forankre seg i veldefinerte og stabile geografiske enheter, kan gi en mer presis og pålitelig representasjon av geografiske forhold.

Min ontologi har som mål å etablere grunnlaget for å skille mellom ulike typer geografiske lokasjoner. Jeg inkluderer en rekke grunnleggende kategorier, slik som land, hav (vannområder), veier og fjell. Disse representerer de fundamentale typene av lokasjoner som er viktige for å forstå geografiske strukturer. Innenfor denne ontologien får hver spesifikke type lokasjon en presis beskrivelse som avgjør dens tilhørighet til en bestemt gruppe av lokasjoner. Et illustrerende eksempel er hvordan land karakteriseres som permanente geografiske områder på landjorden som strekker seg over et definert areal.

Det er også viktig å merke seg at de navngitte lokasjonene jeg inkluderer, er steder som er offisielt anerkjent på landsbasis for å sikre deres inkludering i ontologien. Dette betyr at hver lokasjon som er en del av ontologien, er nøye utvalgt og godkjent som en gyldig geografisk enhet. Denne tilnærmingen sikrer at ontologien ikke bare er omfattende, men også presis og pålitelig når det gjelder å representere geografiske strukturer og deres tilknyttede egenskaper. Her er noen eksempler på noen av de hvordan svarene fra few-shot varierte i forhold til one-shot, og zero-shot.

Hvordan kan denne ontologien forbedres?

Min ontologi fokuserer på å legge grunnlaget for forskjellene mellom ulike typer geografiske lokasjoner, men det er alltid rom for forbedring og videreutvikling. En mulighet for forbedring er å inkludere flere detaljer om hver geografisk lokasjonstype for å gjøre ontologien enda mer

informativ og nyttig. Dette kan inkludere å definere ytterligere egenskaper og karakteristikker for hver lokasjonstype, for eksempel klimatiske forhold, eller demografiske data.

Videre kan ontologien utvides for å omfatte en større variasjon av geografiske elementer. For eksempel kan det være hensiktsmessig å inkludere øyer, elver, innsjøer og andre naturlige formasjoner, samt bygninger, og byer. Ved å utvide ontologien på denne måten, vil den kunne gi en mer omfattende og detaljert representasjon av geografiske fenomener.

8.2 Implementing av kunnskapsgraf for relasjons visualisering

Kunnskapsgrafen er implementert gjennom Python i et rammeverk som heter Dash. Dash er et rammeverk for visualisering og håndtering av data. Opprinnelig var planen å bruke Dash for å visualisere grafen. Imidlertid oppdaget jeg at Dash ikke tillater kunnskapsgrafer. Til tross for dette valgte jeg fortsatt å bruke Dash som webapplikasjonsrammeverket for prosjektet. Jeg utforsket alternativer og fant en løsning ved å bruke en pakke kalt PyVis for å generere kunnskapsgrafen og lagre den som en HTML-fil.

Deretter integrerte jeg denne HTML-filen i Dash-applikasjonens brukergrensesnitt ved hjelp av et HTML iframe-objekt. Dette tillot meg å legge til grafen sømløst i applikasjonen. Ved å utnytte iframe-objektet kunne jeg presentere grafen og dens interaktive funksjoner i Dash-applikasjonen på en simpel måte.

Den resulterende grafen er justerbar både når det gjelder plasseringen av nodene og fremhevingen av relasjonene knyttet til en spesifikk node. Dette gjør det mulig for brukere å utforske og analysere kunnskapsgrafen på en interaktiv og informativ måte, og gir dermed en forbedret opplevelse av datautforskning og visualisering. Se illustrasjon 7.3 for eksempel av visualiseringen.

8.3 Utforskning av prestasjonsutfordringer i prediksjonsalgoritmen

I min undersøkelse av algoritmen for nøyaktighet har jeg oppdaget flere underliggende årsaker til at resultatene muligens ikke lever opp til forventningene. Disse utfordringene skyldes hovedsakelig av kompleksiteten i å modellere, og forstå kontekstuelle relasjoner mellom enheter i datasettet.

Diverse relasjonstyper API'et forsøker å identifisere og klassifisere relasjoner mellom forskjellige enheter, for eksempel lokasjoner eller personer som er basert på forhåndsbestemte parametre. Dersom disse parametrene ikke reflekterer nøye den faktiske kompleksiteten eller variabiliteten i dataene, kan algoritmens evne til å gjøre korrekte forutsigelser svekkes.

Feiltolkning av data Algoritmens avhengighet av nøyaktig dataformat og korrekt tolkning av JSON-strukturer fører til at selv små avvik i dataformatet kan forårsake betydelige forutsigelsesfeil. Denne følsomheten for datakvalitet krever ekstrem presisjon i datatilrettelegging, noe som ikke alltid er mulig i komplekse datasetter. Et eksempel for dette er måten antall relasjoner blir telt opp. Om man teller en enkel relasjon som en unik relasjon, eller om en hel rad blir tolket som en relasjon. Om dataen blir tolket på feil måte, kan dette føre til ekstremt uforventede svar som ikke samsvarer med den faktiske mengden av enkelte relasjoner.

Balansen mellom fullstendig og halvkorrekte forutsigelser Min tilnærming til å evaluere algoritmens presisjon ved å inkludere både fullstendig og halvkorrekte forutsigelser, avdekket en nyansert utfordring. Det viste seg at mange av de 'korrekte' forutsigelsene egentlig bare delvis stemte overens med de faktiske relasjonene, noe som indikerer at algoritmen kan gjenkjenne noen aspekter av relasjonene, men ikke i tilstrekkelig grad for å anses som fullstendig nøyaktig. Bruk av gode vekter er også viktig for tilfeller med halvkorrekte resultater, noe som kan være en veldig kompleks oppgave i seg selv.

Disse utfordringene understreker behovet for videre forskning og utvikling for å forbedre algoritmens robusthet og nøyaktighet. Ved å adressere disse spesifikke problemområdene, kan vi forbedre algoritmens evne til nøyaktighet mer presist og effektivt.

Kapittel 9

Konklusjon

I konklusjonen har dette prosjektet utviklet bruk av avanserte teknologier for naturlig språkbehandling, spesielt GPT-3/4, for å trekke ut og organisere geospatiale data fra ustrukturert tekst. Ved å integrere disse teknologiene med geospatiale ontologier, har jeg utviklet en kunnskapsgraf som effektivt lagrer og avdekker forhold mellom geografiske enheter. Dette har implikasjoner for automatisering i journalistikken, ved å gi journalister verktøy som forbedrer nøyaktigheten og dybden i rapporteringen med geografiske detaljer.

Videre har forskningen demonstrert at ved å utnytte kraften i kunstig intelligens, er det mulig å overvinne noen av de tradisjonelle utfordringene i journalistikken, som raskt å verifisere store mengder stedsbasert informasjon og koble forskjellige datapunkter på meningsfulle måter. Denne kapasiteten er spesielt verdifull i situasjoner som krisejournalistikk, hvor tidsriktig og nøyaktig informasjon er kritisk.

Den utviklede programvareprototypen representerer en praktisk anvendelse som kan brukes direkte i journalistiske arbeidsflyter, og potensielt transformere hvordan redaksjoner fungerer ved å automatisere ekstraksjon og verifisering av geospatiale data. I tillegg har dette prosjektet lagt grunnlaget for fremtidige prosjekter innenfor feltet for automasjonsjournalistikk. Det oppfordrer til videre utforskning i integrasjonen av kunstig intelligens med andre typer ontologier og dataanalyseteknikker, noe som kunne føre til enda mer sofistikerte verktøy.

Ettersom prototypen fortsetter å bli forbedret og testet under forskjellige forhold, kan dens tilpasningsevne og pålitelighet i profesjonelle omgivelser sikres. Denne iterative prosessen vil

ikke bare forbedre funksjonaliteten til verktøyet, men også bidra til den bredere diskursen om etisk bruk av AI i journalistikken. I sin helhet adresserer dette prosjektet ikke bare et eksisterende behov innen journalistikkmiljøet, men åpner også døren til fremtidige muligheter som kan forbedre nyhetsrapporteringens landskap.

Dette prosjektet har gjort fremskritt innen automatisert journalistikk, og tilbyr verdifulle verktøy og innsikter for fremtidige prosjekter. Ettersom jeg fortsetter å utforske og forbedre disse teknologiene, er potensialet for å forbedre nøyaktigheten, hastigheten og dybden av nyhetsrapportering svært lovende for journalistikkfeltet i sin helhet.

Kapittel 10

Takk

Jeg er takknemlig til Bjørnar Tessem for hans veiledning og hjelp, noe som har vært ekstremt hjelpsomt gjennom min master oppgave.
Jeg vill takke mediekonsernet AMedia for deres samarbeid med utdeling av gode og relevante nyhetstekster til masteroppgaven, noe som var til stor hjelp og tidseffektivt.

Roald André Kvarv

Bergen, XX

Bibliografi

Luciano Floridi and Massimo Chiriatti. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines*, 30(4):681–694, December 2020. ISSN 1572-8641. doi: 10.1007/s11023-020-09548-1. URL https://doi.org/10.1007/s11023-020-09548-1. 2.1.1

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity, February 2023. URL http://arxiv.org/abs/2302.04023. arXiv:2302.04023 [cs]. 2.1.3

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, July 2020. URL http://arxiv.org/abs/2005.14165. arXiv:2005.14165 [cs] version: 4. 2.1.3

Yu Zhang, Wenzhou Wu, Qi Wang, and Fenzhen Su. A Geo-Event-Based Geospatial Information Service: A Case Study of Typhoon Hazard. *Sustainability*, 9(4):534, April 2017. ISSN 2071-1050. doi: 10.3390/su9040534. URL https://www.mdpi.com/2071-1050/9/4/534. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute. 2.1.4, 2.2.3

Ashwin Sharma, Disha Devalia, Wilfred Almeida, Harshali Patil, and Aniket Mishra. Statistical Data Analysis using GPT3: An Overview. In *2022 IEEE Bombay Section Signa-*

ture Conference (IBSSC), pages 1–6, December 2022. doi: 10.1109/IBSSC56953.2022.10037383. 2.1.5

Xin Wang, Wei Gu, Danielle Ziebelin, and Howard Hamilton. An ontology-based framework for geospatial clustering. *International Journal of Geographical Information Science*, 24 (11):1601–1630, November 2010. ISSN 1365-8816. doi: 10.1080/13658811003702147. URL https://doi.org/10.1080/13658811003702147. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/13658811003702147. 2.2.2

Christophe Claramunt. Ontologies for geospatial information: Progress and challenges ahead. *Journal of Spatial Information Science*, (20):35–41, June 2020. ISSN 1948-660X. URL https://josis.org/index.php/josis/article/view/111. Number: 20. 2.2.2

IBM. What Is a Knowledge Graph?, March 2024a. URL https://www.ibm.com/topics/knowledge-graph. 2.3.1

W3C. RDF 1.1 Concepts and Abstract Syntax, 2014. URL https://www.w3.org/TR/rdf11-concepts/. 2.3.2

Xuan Guo, Haizhong Qian, Fang Wu, and Junnan Liu. A Method for Constructing Geographical Knowledge Graph from Multisource Data. *Sustainability*, 13(19):10602, January 2021. ISSN 2071-1050. doi: 10.3390/su131910602. URL https://www.mdpi.com/2071-1050/13/19/10602. Number: 19 Publisher: Multidisciplinary Digital Publishing Institute. 2.3.3

Salvatore T. March and Gerald F. Smith. Design and natural science research on information technology. page 16, 1995. URL https://mitt.uib.no/courses/38903/files/folder/Designforsking?preview=4945334. 4.1, 4.1.1

Balakrishnan. Geospatial Data Validation Procedure and Techniques. March 2019. doi: 10.15515/iaast.0976-4828.10.1.148153. 4.2.1

The European Union and the Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016. 4.2.2

Betsie Estes. Geolocation—The Risk and Benefits of a Trending Technology, September 2016. URL https://www.isaca.org/resources/isaca-journal/issues/2016/volume-5/geolocationthe-risk-and-benefits-of-a-trending-technology. 4.2.2

Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. GPT3-to-plan: Extracting plans from text using GPT-3, June 2021. URL http://arxiv.org/abs/2106.07131. arXiv:2106.07131 [cs]. 5.2

Mark Coeckelbergh. Artificial Intelligence: Some ethical issues and regulatory challenges. *Technology and Regulation*, 2019:31–34, May 2019. ISSN 2666-139X. doi: 10.26116/techreg.2019.003. URL https://techreg.org/article/view/10999. 5.3

Anneke Zuiderwijk, Yu-Che Chen, and Fadi Salem. Implications of the use of artificial intelligence in public governance: A systematic literature review and a research agenda. *Government Information Quarterly*, 38(3):101577, July 2021. ISSN 0740-624X. doi: 10.1016/j.giq.2021.101577. URL https://www.sciencedirect.com/science/article/pii/S0740624X21000137. 5.3

Amazon. What is Prompt Engineering? - AI Prompt Engineering Explained - AWS, 2024. URL https://aws.amazon.com/what-is/prompt-engineering/. 6.1.1

Ivan Y. Tyukin, Alexander N. Gorban, Muhammad H. Alkhudaydi, and Qinghua Zhou. Demystification of Few-shot and One-shot Learning, May 2021. URL http://arxiv.org/abs/2104.12174. arXiv:2104.12174 [cs, math, stat]. 6.1.2

W3C. SPARQL 1.1 Query Language, 2024a. URL https://www.w3.org/TR/sparql11-query/. 6.2.4

Open Geospatial Consortium. GeoSPARQL - A Geographic Query Language for RDF Data, 2024. URL https://www.ogc.org/standard/geosparql/. 6.2.4

Chris Hoffman. What Is a CSV File, and How Do I Open It?, April 2018. URL https://www.howtogeek.com/348960/what-is-a-csv-file-and-how-do-i-open-it/. 6.2.5

W3C. RDF 1.1 Turtle, 2024b. URL https://www.w3.org/TR/turtle/. 6.2.5

Dave Bergmann. What is fine-tuning, March 2024. URL https://www.ibm.com/topics/fine-tuning. 7.1.3

OpenAI. Fine-Tuning, 2024. URL https://platform.openai.com/docs/guides/fine-tuning. 7.1.3

IBM. What is Overfitting?, March 2024b. URL https://www.ibm.com/topics/overfitting. 8.1.1