Høgskulen på Vestlandet

BACHELOROPPGAVE

Tolking av PDF-notesett og gjenkjenning av stemmer

Interpretation of PDF sheet music and recognition of parts

Martin Tunge Sterri

Lars Martin Taraldset

Halldor Broddi Thorsteinsson

Dataingeniør
Institutt for data- og realfag
Fakultet for ingeniør- og naturvitenskap
Veileder – Atle Birger Geitung
Innleveringsdato 22.05.2023

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel:	*Dato:*
Tolking av PDF-notesett og gjenkjenning av stemmer.	22.05.2023
Forfatter(e):	*Antall sider u/vedlegg: 60*
Martin Tunge Sterri, Lars Martin Taraldset, Halldor Broddi Thorsteinsson	*Antall sider vedlegg: 89*
Studieretning:	*Antall disketter/CD-er:*
Dataingeniør	0
Kontaktperson ved studieretning:	*Gradering:*
Atle Birger Geitung	Ingen
Merknader:	
Koden i prosjektet er konfidensiell. Tilgang under sensur finnes i Systemdokumentasjon-vedlegget.	

Oppdragsgiver:	*Oppdragsgivers referanse:*
Styreportalen AS	
Oppdragsgivers kontaktperson:	*Telefon:*
Stian Sømoen	913 19 131

Sammendrag:

Dette prosjektet handler om å lage en API-løsning for Styreportalen. API-et skal gjøre det lettere å legge inn nye notesett inn i Syreportalen sitt notearkiv. API-et bruker Google Vision API, en maskinlæringsmodell og en algoritme for å finne relevant informasjon fra notesett.

Stikkord:

Maskinlæring	Notesett	API

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

Forord

Vi ønsker å uttrykke vår takknemlighet til alle som har bidratt til at vi kunne fullføre dette arbeidet.

Spesielt vil vi takke Stian Sømoen fra Styreportalen. Hans samarbeid har vært viktig i utviklingen av vårt prosjekt. Hans engasjement for prosjektet har hjulpet oss å bringe API-et til liv. Vi er veldig takknemlig for hans tid og tålmodighet gjennom hele prosessen.

Vi ønsker også å uttrykke vår takknemlighet til vår veileder, Atle Birger Geitung. Hans faglige veiledning, kritiske innsikt, og oppmuntrende støtte har vært viktig for prosjekt. Hans kontinuerlige støtte og konstruktive tilbakemeldinger har hjulpet oss å forstå og navigere de utfordringene vi møtte under utviklingen.

Vi er også takknemlige for hverandre, fordi uten samarbeid, kommunikasjon og felles innsats, ville dette prosjektet ikke ha vært mulig. Vi har lært enormt fra hverandre, og det har vært en glede å jobbe sammen på denne reisen.

Til slutt, ønsker vi å takke våre venner og familier for deres kontinuerlige støtte og tålmodighet gjennom dette semesteret. Deres forståelse og oppmuntring har betydd mye for oss.

Bachelorgruppen,

- Martin Tunge Sterri, Lars Martin Taraldset og Halldor Broddi Thorsteinsson

Ordbok

Ord	Forklaring
Adobe Acrobat Pro	Verktøy for PDF
Agile	Arbeide smidig som gjør at er altid mulighet for endringer
Algoritme	En serie av operasjoner som skal utføres. Det brukes for å løse problemer.
Algoritmebasert løsni	Bruker algoritmer for å løse oppgaven.
API	Application porgramming interface. API gir tilgang for programmer/datamaskiner å kommunisere med hvem andre.
API-endepunkt	Er punkt der API-et koble til programvaren for å få komminikasjon
Applikasjon	Data program som løse oppgave eller gi infromasjon og tjeneste for bruker.
Arrangør	Person som adaptere stykket. Endre for å passe til pesifikke behov.
Blocks	Øverste nivå av infromasjon i JSON-fil.
Bucket	Oppevaring plass for filer.
Concierce	Brukes her som personer som arbeider kun for en annen person for å gjør det så bra for han som mulig.
Delstring	Streng som inneholder en del av en annen streng.
Digitalisere	Er å bruke teknologi til å forbedre, forenkle og/eller fornye.
Dyplære	Er en underkategori av maskinlæring som bruker kunstige nevrale nettverk med mange lag for å modellere og forstå komplekse mønstre i data.
Fastai	Rammeverk for deep lære
Firebase	Utviklings platform laget av Google
Flagging	Forteller brukeren om det er noge mer som sendes tilbake.
Flask-rammeverk	Python rammeverk gor å lage webapplikasjoner
Forsider	Side i notesett som er fremst og inneholder infromasjon som ikke er noter.
Gantt-diagram	Diagram som viser porsjekt planen.
Google Cloud Platform	Google sin skytjeneste
Google Vision API	Inneholder allrede trente maskin lære modeller.
Instrumenteliste	Instrumentliste er listen bruk for å holde data på instrumenter som kan være på noteark.
Iterativt	Dele prosjektet ned i flere små prosjekter.
Jaro-Winkler-distanse	Måle enhet for å måle forskjell på 2 strenger.
Java	Programmering språk.
JavaScript	Programmering språk.
JSON	JavaScripts Object Notation, er enkel tekstabasert standart for å formutere meldinger.
Komponist	Han som skrev stykket.
Maskinlæring	Lære datamaskiner å lære av data og få erfaring.
Musikkorps	Gruppe av person i et lag/forening som spiller i musikk lag.
MVP	Minimum valuble product, er en praksis fra The Lean Startup å lage så enkelt produkt som mulig så fort så mulig.
Nanonets	Maskin lære modell
Noteark	Noteark brukes over en enkel side i et notesett.
Notearkiv	Er en digital løsing for lagring av noter.
Notesett	Notesett er alle arkene som er gitt ut for en sang som skal spilles. Samling av flere noteark.
OCR	Optical Character Recognition er navnet over metode der datasystem lerser tegn teks i et bilde. Det gjøres til digital representasjon.
Oppdragsgiver	Oppdragsgiver er styreportalen.
Paragraph	Midt nivå av infromasjon i JSON-fil.
Partitur	Sammenstilling av alle stemmene på et enkelt noteark.
PDF	Portable Document Format er filformat for utveksling av digitale dokumenter.
Programmeringspråk	Programmering språk er en betagnelse over språk som skrive programmer til datamaskiner.
Python	Programmering språk.
ResNet	Rammeverk for deep lære.
Risiko matrise	Matrise som inneholder risiko nivåer.
Risiko tabell	tabell som inneholder risikoen og hvordan dem skal unngå.
Splitting	Dele notesette opp i mindre deler der hver del har hver sin stemme.
Stemme	Er notesiden til et instrumentet skal spilles.
Styreportalen	Styreportalen er firmaet som har gitt ut denne oppgaven
The Lean Startup	Arbeidsmetotikk.
Tittel	Navnet på stykket.
TUTTI	Medlemsapp fra Styreportalen.
Utforminsprinsipper	Er standard for kode sånn at enklere er å vedlikeholde eller videreutvikle.
Word	Lavaste nivå av infromasjon i JSON-fil

Tabell 1: ordbok

INNHOLDSFORTEGNELSE

1 Innledning

1.1 Prosjekteier

Styreportalen AS er et firma med et mål om å forenkle drift av lag og foreninger. Deres hoved er og lager løsninger for musikkorps. Dette gjøres ved bruk av flere forskjellige applikasjoner som å sette opp kalender/terminlister, styrearbeid, eiendeler og et notearkiv. Styreportalen skreddsur sine verktøy for at den daglige driften blir enklere ved å ta i bruk medlemsappen TUTTI. Det TUTTI gjør er at den samler all kommunikasjon og informasjon til en plattform (Styreportalen, ingen dato).

Styreportalen har nærme 2000 organisasjoner som aktivt bruker deres løsninger og er i konstant vekst (Styreportalen, ingen dato).

1.2 Kontekst

En viktig resurs for musikkorps er notesettene deres. Det er nødvendig for musikkorpsene å ha fysiske notearkiv. Disse arkivene kan bli store og vanskelig å holde orden på. For å dele ut notene, må det lages kopier. I dag er det et skifte mot digitale løsninger der musikkorps ikke lenger kjøper fysiske versjoner, men PDF-filer. Det er også flere musikkorps som ønsker å digitalisere sine fysiske arkiv. Dette er utgangspunktet for at Styreportalen har utviklet et produkt som heter *Notearkiv*. Dette er et digitalt arkiv hvor det er mulig å lagre notesett som PDF-filer.

En viktig funksjon som arkivet har, er knyttet til utdelingen av noter. Hver spiller vil ha en spesifikk stemme som de skal spille. Stemmen er notesidene til et instrument som skal spilles. Flere kan spille samme instrument, men med forskjellige stemmer. Når notesett skal deles med de som skal spille stykke, sendes dette via e-post eller TUTTI. Når dette gjøres, skal spillerne kun får tildelt sine relevante stemmer. Bakgrunnen til at de ikke kan dele ut hele settet, er at det finnes strenge regler knyttet til deling av noter. Derfor er notesettet nødt til å bli delt opp i stemmer for å kunne bruke *Notearkiv*.

1.3 Motivasjon

For å kunne benytte seg av *Notearkiv*, må notesettet bli splittet inn i stemmer. Løsningen som brukes i dag er manuell, og krever at brukeren tildeler hver side en stemme. Denne prosessen tar ca. 2 minutter per notesett. Det finnes organisasjoner som har flere tusen notesett. Arbeidet som må gjøres for å legge inn disse notesettene vil ta stor tid. Det å forenkle prosessen med å legge inn nye sett, er noe flere av brukerne ønsker seg, ifølge Styreportalen.

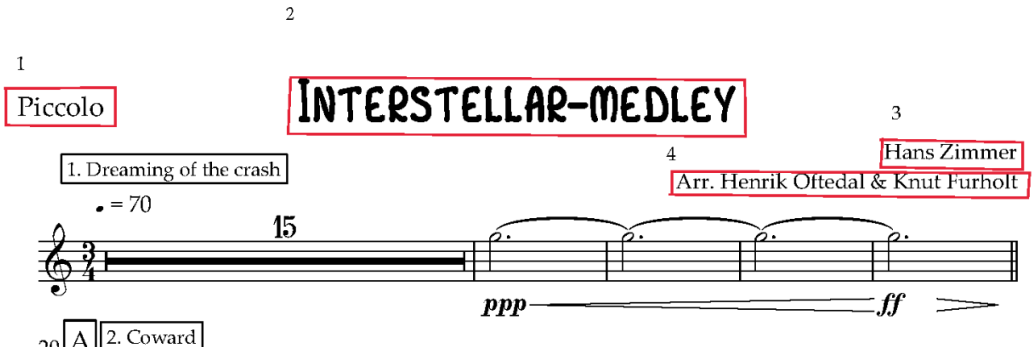
Styreportalen ønsker derfor å automatisere deler av denne prosessen. Hvilken stemme en side er knyttet til og annen informasjon om notesettet, står på sidene. Denne informasjonen

må i dag skrives inn manuelt. Om det er mulig å lage en løsning som kan hente ut denne informasjonen fra PDF-ene, kan det gjøre det mer brukervennlig å legge inn nye notesett.

Ut fra oppdragsgiveren sine undersøkelser, finnes det ikke verktøy som de kan benytte for å gjøre dette. Det finnes verktøy for å hente ut informasjon fra dokumenter som Document AI (*Document AI*, ingen dato), men siden strukturen kan være forskjellige, gir disse trolig ikke ønskede resultater for Styreportalen.

1.4 Problembeskrivelse og mål

Styreportalen ønsker et Application Programming Interface (API) som de kan bruke til å finne hvilke stemmer som hører til hver side. De ønsker også å hente ut annen informasjon som tittel, komponist og arrangør, slik at dette ikke trengs å gjøres manuelt. Denne informasjonen finnes i øverste del av notearkene (Figur 1).



Figur 1: Eksempel på data på notesett. Boks 1 - stemmen, boks 2 - tittel, boks 3 - komponist, boks 4 – arrangør (Zimmer, Oftedal og Furholt, ingen dato)

API-et skal returnere en JavaScript Object Notation-fil (JSON-fil) med informasjonene som blir hentet ut.

Målet er å lage et API som har høy nok nøyaktighet til at informasjonen som hentes ut kan brukes videre i automatisering eller forenkling av arbeidet med å legge inn nye notesett. API-et bør gi riktig resultat i minst 80-90% av notesettene. Et delmål vil være å flagge informasjon som kan være feil.

1.5 Oppbygging av rapporten

Rapporten begynner med en innledning som forteller om bakgrunnen til hvorfor prosjektet gjennomføres. Innledningen inkluderer i tillegg kontekst, motivasjon og en problembeskrivelse med mål.

Kapittel 2 er prosjektbeskrivelsen. Her finnes detaljer rundt grunnlaget som prosjektet bygges på. Kapittelet inkluderer en beskrivelse av tidligere arbeid, litratur og andre ressurser som notesett. Det gis også en beskrivelse av krav fra oppdragsgiveren og avgrensinger i oppgaven.

Kapittel 3 beskriver flere valg som ble tatt i startfasen av prosjektet. Kapittelet inkluderer ting som valg av løsning, verktøy og fremgangsmåte. Prosjektmetodikk og evalueringsplan har blitt diskutert her. Disse blir viktige for gjennomføringen av prosjektet.

Kapittel 4 beskriver løsningen. Kapittelet er bygget opp på en måte som følger hoved flyten til løsningen. Andre aspekter med det som har blitt utviklet diskuteres underveis i egne underkapitler.

Kapittel 5 beskriver evalueringen og resultatene fra denne. Prosjektgjennomføringen blir også beskrevet her.

Kapittel 6 reflekterer rundt ulike aspekter av løsningen og prosjektgjennomføringen.

Kapittel 7 er konklusjonen. Her presenteres også mulig videre arbeid.

Kapittel 8 er figur- og tabell-listene. Listene er en oversikt over de ulike figurene og tabellene som finnes i rapporten.

Kapittel 9 er referanselisten.

Kapittel 10 gir en kort oversikt over innholdet i vedleggene.

2 Prosjektbeskrivelse

I denne delen av rapporten, skal bakgrunnen til arbeidet utvides. Prosjektet sin praktiske bakgrunn, tidligere arbeid, initiale krav fra oppdragsgiver, initiell løsnings-ide relatert til kravene i starten, må prosjektet avgrenses, hvilke ressurser som er benyttet og til slutt litteratur bruk.

2.1 Bakgrunn

En lignende oppgave har blitt jobbet med tidligere. Bachelorrapporten «Utvikling av kategoriseringsverktøy for PDF-notesett» fra våren 2022, fokuserer på en lignende løsning for Styreportalen. Selv om oppgaven har blitt gitt tidligere, mente Styreportalen at løsningen ikke var god nok til å kunne brukes. Dette førte til at oppgaven måtte gjøres på nytt.

2.1.1 Tidligere arbeid

Løsningen som ble utviklet av den tidligere gruppen, var ikke god nok til at oppdragsgiver kunne ta den i bruk. I rapporten kommer det frem at de oppnådde et resultat der de klarte å finne riktig stemme 77% av sidene. Denne andelen representerer antall riktige sider, og ikke riktige notesett. På notesett er løsning langt dårligere (Engevik og Engevik, 2022).

Kravene som ble presentert er ikke de samme. Rapporten peker mot at oppdragsgiver ønsker en mer helhetlig løsning for splitting, ikke bare et API. Oppdragsgiver fjernet i tillegg en del av oppgaven som går på at API-et skal returnere et standardisert stemmenavn. Det at API-et har forskjellige krav til hva som skal returneres gjør at målingene ikke kan sammenlignes.

I rapporten kommer det også frem at oppdragsgiveren i det tidligere prosjektet også hadde flere krav til verktøy som skulle benyttes. I denne gjennomføringen har ikke oppdragsgiveren hatt krav knyttet til språk, tjenesteløsning og Optical Character Recognition (OCR). Selv om mange av de samme verktøyene har blitt brukt, har det blitt tatt egne valg. Disse er begrunnet i kapittel *3.3 Valg av verktøy*.

Det er flere aspekter med løsningen som gjør den mindre egnet som et utgangspunkt for videreutvikling. De forskjellige kravene skaper problemer med å bruke deres løsning som grunnlag i dette prosjektet. En del av prosessen blir utført av klientsiden, selv om det i den nye løsningen passer bedre på tjenestesiden. Blant annet må klienten sende kall til to forskjellige API-er. Det første kallet brukes til å kjøre OCR-verktøyet og det andre til algoritmen som gruppen har laget. Disse er skrevet i forskjellige programmeringsspråk, noe som gjør det utfordrende å gjøre dem om til ett API.

Kodens struktur gjør det også utfordrende å bygge videre på den tidligere løsningen. Det finnes utformingsprinsipper som skal gjøre kode mer forståelig og lettere å vedlikeholde eller videreutvikle. Flere av disse prinsippene går ut på å dele ansvaret for ulike deler av

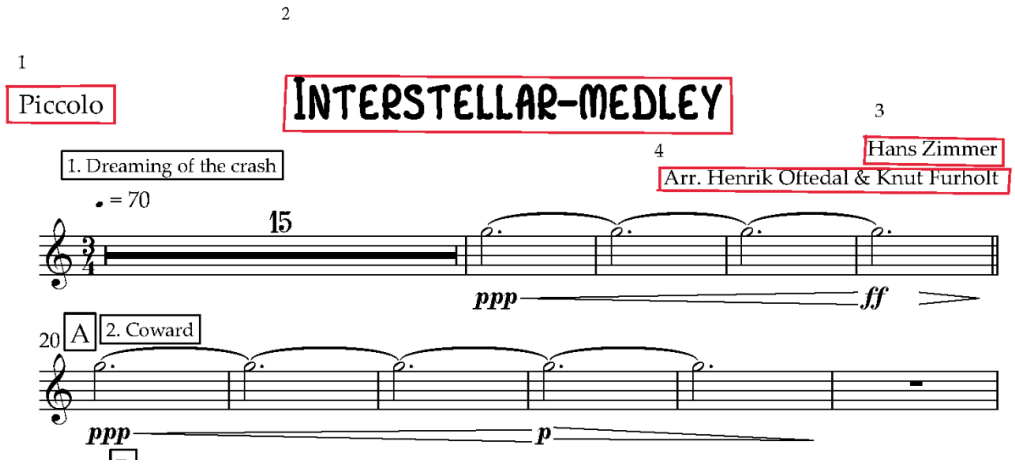
programmet opp i forskjellige moduler/klasser, slik at hver modul blir oversiktlig og det blir lett å bytte ut med noe nytt (Boccara, 2021). Algoritme-API-et består av tre Python-filer. Den første er API-endepunktet, det andre er algoritmen og den tredje er en liste med ulike stemmer. Algoritmen består av et dokument på over 300 linjer, der rundt halvparten hører til én funksjon. All informasjonen som algoritmen skal finne, blir utført i samme løkke. Det er vurdert at flere av utformingsprinsippene ikke ble fulgt, noe som tyder på at koden er vanskelig å arbeide med. Ekstraarbeidet med å benytte deres kode ses på som større enn det arbeidet som kommer med å begynne på nytt.

Selv om koden ikke benyttes, finnes det nyttig erfaring som kan tas fra prosjektene. En del av dette handler om utfordringer knyttet til utformingen av notesettene. Rapporten har også vært nyttig i å få en dypere forståelse for oppgaven i en tidlig fase av prosjektet. Det reflekteres også over muligheter for å benytte maskinlæring i det videre arbeidet.

2.1.2 Initielle krav

Styreportalen har hatt begrenset med krav når det kommer til flere aspekter med løsningen. Blant annet har det vært begrenset med krav knyttet til valg av språk, tjenesteløsning og andre verktøy som blir aktuelle.

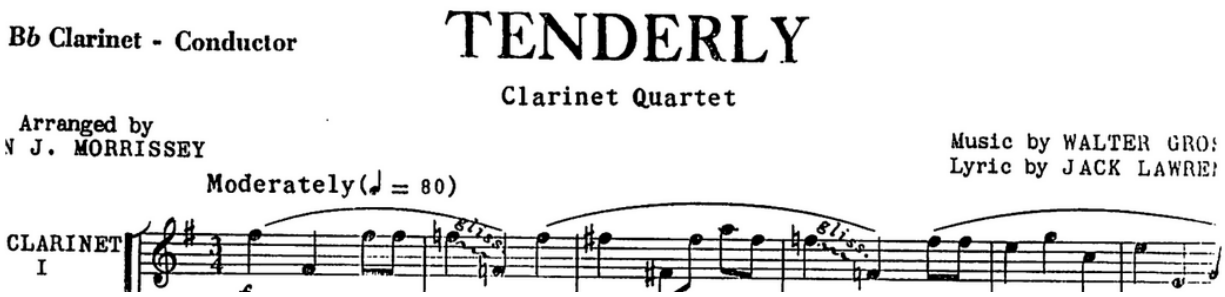
Styreportalen ønsker en API-løsning som returnerer en JSON-fil med info knyttet til notesettet. JSON-filen må inneholde informasjon om hvilken stemme hver side inneholder. Den bør også inneholde tittel, komponist og arrangør. Tidlige krav finnes i Kravdokumentet (Sterri, Taraldset og Thorsteinsson, 2023a).



Figur 2: Eksempel på toppen av en normal noteside. Bokser: 1 – stemmen, 2 – tittel, 3 – komponist, 4 – arrangør.(Zimmer, Oftedal Furholt, ingen dato)

Oppdragsgiver har vært tydelig på at det ikke er mulig å få til en løsning som fungerer i alle tilfeller. Det viktigste vil være å tilpasse programmet slik at det fungerer på de fleste notesettene med «standard» oppsett. Oppsettene som i dette tilfellet regnes som standard har stemmen i øvre venstre hjørne, tittel befinner seg i midten og på høyre side finnes komponist og arrangør. Komponist ligger som regel øverst. Arrangør har ofte «Arr.» eller «Arrangør» først på linjen (Figur 2).

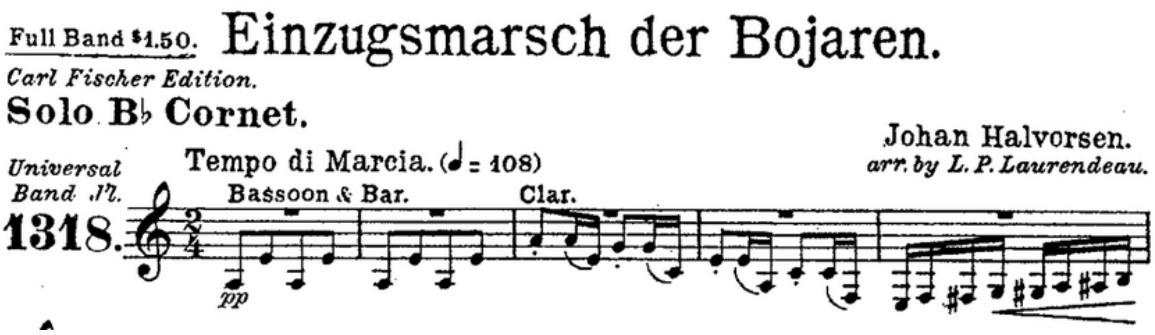
Det finnes notesett som ikke er satt opp slik. Det kan være et problemer at ikke all informasjon er på den forventede plassen som i Figur 3, hvor arrangøren er på venstre side.



Figur 3: Eksempel på side der informasjonen ikke er på standardposisjonen. Her er arrangøren plassert på venstre side under stemmen. (Gross, Lawrence og Morrissey, ingen dato)

En utfordring som kan oppstå er at stemmen havner på to linjer. Selv om den er over to linjer, er det viktig å få med seg all informasjonen (Figur 4). En annen utfordring er at det kan være tekst på venstre side som ikke tilhører stemmen (Figur 5). Det kan være en utfordrende å skille stemmen og annen tekst når stemmen skal finnes.

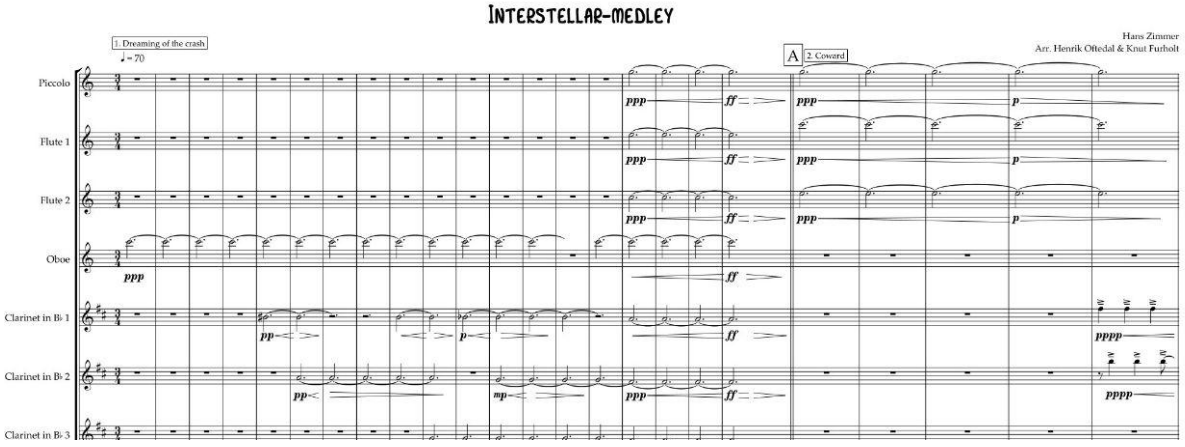
Figur 4: Eksempel på side der stemmen går over to linjer.(Bernie et al.*, ingen dato)*



Figur 5: Eksempel på side der det er tekst rundt stemmen som ikke skal være med. (Halvorsen og Laurendeau, ingen dato)

En annen utfordring er knyttet til at ikke alle sider hører til en stemme. Enkelte notesett inneholder forsider og sider med tekst (biografier osv.). Disse sidene bør skilles ut. En annen type side er partitursiden. Et partitur i notesett er en sammenstilling av alle stemmene på et enkelt noteark (Nagelhus, 2008). Utseende på disse sidene skiller seg fra normale stemmesider. Navnet på de ulike stemmene står på venstre side. Det finnes i noen

tilfeller tekst som tilsier at dette er et partitur. For eksempel kan det stå «score». I flere tilfeller står ingen tekst (Figur 6).



Figur 6: Eksempel på toppen av en partiturside. (Zimmer, Oftedal og Furholt, ingen dato)

Flere av kravene som ble nevnt i starten av prosjektet var knyttet til stemmene og hvordan det skulle formateres. En enkelt stemme, kan skrives på flere måter. Eksempler på dette er at «cornet 3», «cornet iii» og «3. kornett» refererer til samme stemme. Oppdragsgiver hadde et ønske om alle disse skulle returnere den samme teksten. Mer om dette finnes i Visjonsdokumentet (Sterri, Taraldset og Thorsteinsson, 2023d). Denne delen ble fjernet underveis i prosjektet ettersom oppdragsgiver fant ut at det er mer hensiktsmessig å gjøre denne prosesseringen på deres side. Dette skjedde før utviklingen av den delen av prosjektet hadde begynt.

Løsningen som ble utviklet, skal brukes i prosessen med å legge til nye notesett i *Notearkiv*. Tiden den bruke vil påvirke brukervennligheten. Selv om dette er et aspekt, er det viktig at løsningen er nøyaktig nok til at den kan brukes i automatiseringen. Oppdragsgiver har vært tydelig på at tid i utgangspunktet ikke bør være en prioritering.

2.2 Avgrensninger

For at et prosjekt ikke skal bli for stort, er det viktig å sette avgrensinger som gjør oppgaven mer konkret og oppnåelig. En av aspektene med det tidligere prosjektet, er knyttet til klientsiden. Det ble laget en demonstrasjonsside som ble brukt til testing av applikasjonen. Strukturen på prosjektet viser at denne siden inkluderte deler av løsningen. Det var altså funksjonalitet i demonstrasjonssiden utover det å sende inn en PDF-fil. Dette skulle senere implementeres i Styreportalen løsning (Engevik og Engevik, 2022). Det er et inntrykk at klientsiden er en større del av løsningen. Mangelen på avgrensing i den tidligere oppgaven er noe av grunnen til at det kan være vanskelig å jobbe videre med deres løsning.

Det har dermed vært viktig å være tydelig i avgrensingen til prosjektet. Oppgaven var avgrenset til API-et. Oppdragsgiverens ønske var et API som de kunne bruke i

arkivløsningen sin. Det var dermed ikke behov for å utvikle en klientside. Selv om det ikke er behov for det, har det blitt laget en enkel side som kan benyttes til å demonstrere hvordan API-et fungerer.

Det ble også gjort avgrensinger i hvilke PDF-filer programmet skal akseptere. Oppdragsgiver har i utgangspunktet vært positiv til å gjøre flere avgrensinger. Blant annet har det vært snakk om at programmet ikke trenger å kunne skille mellom partitur-sider og sider med stemmer. Løsningen ville blitt at disse skulle lastes opp som separate filer. Ikke alle avgrensinger som oppdragsgiver har nevnt, har blitt fullt. Dette kommer av at flere av problemene som oppdragsgiver har sett, har blitt vurdert som mulig å løse uten å begrense hvilke PDF-filer programmet klarer å tolke. Det er fremdeles enkelte ting løsningen ikke tar hensyn til. Dette inkluderer PDF-filer der sidene er rotert feil og PDF-filer der det er to sider per ark.

2.3 Ressurser

Tilgangen til data utgjorde en viktig ressurs under utviklingen av løsningen. Dataen som ble brukt var notesett som PDF-filer og disse var avgjørende for hvilke oppsett som løsningen ble tilpasset.

En begrensing i prosjektet har vært mengden data. Det er strenge regler knyttet til deling av noter. Dette gjorde at selv om *Notearkiv* har flere hundretusen notesett, har det ikke vært mulig å få tilgang til alle disse. Notesettene som har blitt brukt kommer fra oppdragsgiver. Det har blitt gitt tilgang til ca. 70 notesett av ulik kvalitet.

Det finnes muligheter for å hente flere notesett utover det oppdragsgiver gir. I digitale arkiver finnes det notesett under Creative Commons-lisens, som kunne blitt brukt under utviklingen. Utfordringen her er knyttet til formatet på notesettene. Et eksempel på dette er «BandMusic PDF Library». Her finnes originalversjonene som er fra 1927 eller eldre, og en ny versjon. De nye versjonene ser ut til å ha lignende oppsett (BandMusic PDF Library, ingen dato). En mulig utfordring vil være at disse ikke er representative for notesettene som settes i Notearkiv. Det kan gjøre løsningen dårligere. Disse ble dermed ikke brukt.

2.4 Litteratur om problemstillingen

Det har vært utfordrende å finne litteratur knyttet til problemstillingen. Det finnes litteratur for deler av prosjektet.

For maskinlæring har det blitt tatt utgangspunkt i «Deep learning for coders with fastai and PyTorch: AI applications without a PhD». Boken gir grunnlag til hvordan en kan bruke fastai-rammeverket for å bygge og trene dyplæringsmodeller. Innenfor maskinlæring er dyplære et godt verktøy for bildeklassifisering (Howard, Gugger og Chintala, 2020).

En annen bok som har blitt brukt til maskinlæring, er «Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent

Systems». Boken tar utgangspunkt i fundamentene i trening av modeller i maskinlæring. Boken tar utgangspunkt i modeller som ikke er knyttet til dyplære. Selv om den ikke omfatter dyplære, er flere av ideene også nyttige innenfor dette temaet (Géron, 2019).

«Den store musikkordboka» av Lorents Aage Nagelhus ble i utgangspunktet brukt til forklaring av flere av begrepene innenfor domenet (Nagelhus, 2008).

3 Design av prosjektet

I denne delen av rapporten, blir designet av prosjektet beskrevet. Det blir beskrevet alternativer for mulige løsningsorienterte tilnærminger og oppnå målene som ble lagt fram tidligere i rapporten. Hvilke alternativer som ble valgt og hvorfor og hvilke verktøy som ble brukt beskrives. Metodikk, risikoanalysen og evalueringsplanen for dette prosjektet blir framstilt.

3.1 Forslag til løsning

Deler av flyten til programmet ble tydelig i en tidlig fase. Et API-endepunkt tar imot en PDF-fil. Denne filen sendes gjennom en OCR for deretter å finne hvilken del av teksten som hører til stemmen, tittelen, komponist og arrangør. En OCR er et verktøy som brukes for å finne tekst i bilder. Det finnes diverse muligheter til å gjøre prosessering av PDF-filen før den sendes til OCR-en. Det kan blant annet være aktuelt å filtrere bort sider, slik at det bare er sider med nyttig informasjon som hentes ut. Dette vil i utgangspunktet ikke påvirke det endelige resultatet, og blir dermed ikke et hovedfokus.

Hovedfokuset var å identifisere stemmen, tittel, komponist og arrangør i det OCR-en returnerte. Her finnes det ulike tilnærminger som kan benyttes for å løse dette problemet.

3.1.1 Algoritmebasert

En regelbasert tilnærming kan brukes for å finne ut hvilken tekst som inneholder hvilken informasjon. OCR-en kan gi relevant informasjon om tekstens posisjon og dens størrelse. Siden mange av notesettene har lignende struktur, kan visse regler benyttes for å avgjøre hvilken informasjon som tilhører hvilke kategorier. Denne fremgangsmåten blir videre omtalt som en algoritme.

Det tidligere prosjektet baserte arbeidet sitt på en slik tilnærming. Ved å ha en database over mulige stemmer, gjennomførte gruppen et søk i databasen for å se om de fant den ønskede stemmen eller et synonym. Dette resulterte i korrekt identifisering i 77% av tilfellene. For identifisering av tittelen, ble det valgt å returnere den største teksten på siden som tittel, som ga riktig identifisering i 72% av tilfellene. Arrangøren ble funnet ved å lete etter linjer som begynte på «arr» og ga riktig resultat 58% av tilfellene. Komponisten ble funnet som teksten øverst i høyre hjørne. Dette ga riktig resultat 59% av gangene (Engevik og Engevik, 2022).

Resultatene over indikerer på at det er mulig å utvikle en løsning som baserer seg på algoritmer. Det er usikkerhet knyttet til hvor nøyaktig det er mulig å utvikle den. Hvilke typer feil algoritmen gjør, beskrives ikke i rapporten (Engevik og Engevik, 2022). Dette gjør det vanskelig å lage hypoteser på hva som kan gjøre algoritmen bedre.

3.1.2 Maskinlæringsbasert

Et annet alternativ som har blitt nevnt er muligheten til å bruke maskinlæring (Engevik og Engevik, 2022). Modellene kan benytte seg av sammenhenger og egenskaper som er vanskelig å oppdage for et menneske. Dette skaper muligheter til å lage modeller som er bedre enn det som er mulig å lage selv (Géron, 2019).

Det finnes flere muligheter for hva maskinlæring kan benyttes til. Blant annet er det mulig å trene en modell for bildegjenkjenning som kan brukes til å identifisere hvilke sider som er partitur, normal noteside eller andre sider som forside og beskrivelse. Dette kan kombineres med en algoritmebasert løsning.

En annen mulighet der maskinlæring kan benyttes, er å trene en modell som kan identifisere stemmene, tittel, komponist og arrangør. SpaCy er et «Open-Source»-bibliotek i Python som brukes i en rekke oppgaver innen Natural Language Processing (NLP), inkludert Named Entity Recognition (NER). NER-modeller kan brukes til å identifisere deler av en tekst, som for eksempel hvilken del av teksten som refererer til personer eller steder. SpaCy tilbyr forhåndstrente NER-modeller som er trent på store språkbaserte datasett. Disse modellene er tilpasset slik at de passer til en spesifikk type tekst og spesifikke oppgaver (Explosion, ingen dato). I dette prosjektet hadde det være mulig å trene modellen til å identifisere stemme, tittel, komponist og arrangør ut ifra teksten funnet ved hjelp av OCR-en.

Selv om maskinlæring har potensial i en slik oppgave, finnes det utfordringer som kan gjøre det vanskelig å trene en modell til dette formålet. En av hovedutfordringene knyttet til en slik løsning vil være innhenting av data. Som nevnt i *ressurser* er tilgangen til data, altså notesett, en begrensing i dette prosjektet. Det er vanskelig å si hvor mye data som trenges for å lage en god modell. Er det for lite data, er det en mulighet for at modellen blir overtilpasset på det datasettet modellen trenes på. Kommer det da inn nye sett, med nye oppsett eller på nye språk, kan modellen ha utfordringer med å predikere disse notesettene (Géron, 2019).

En annen mulig utfordring er knyttet til usikkerheten rundt hvorfor en modell velger det den velger. Dette er spesielt utfordrende ved bruk av dyplære. Det er mulig å få modeller som gjør feil med stor sikkerhet, uten at det er lett å vite hva som gjør at modellen kommer til denne konklusjonen (Géron, 2019).

3.1.3 Diskusjon av alternativene

Både en algoritmebasert og en maskinlærings basert tilnærming har utfordringer knyttet til usikkerheten om hvor god løsningen kan bli. Som resultat av det, er det nødvendig å teste dem ut i en tidlig fase av prosjektet. Dette vil redusere risikoen for å velge en utilfredsstillende løsning.

For å anvende maskinlæring i denne prosessen, er det nødvendig å markere hva som er ønsket at

skal predikere. Dette har medført ekstra arbeid, men dette har likevel være verdifullt for evalueringen av en algoritmebasert modell. Å arbeidet med en løsning som kan brukes til markering og testing av modellen, har derfor være nyttig uavhengig av hvilket alternativ som velges.

Gitt at en algoritmebasert metode allerede har blitt forsøkt, var det størst usikkerhet knyttet til en maskinlæringsbasert tilnærming. Selv om maskinlæring ble sett på som å ha størst potensial, var usikkerheten knyttet til den så stor at det ikke var mulig å velge vekk en algoritmebasert modell.

3.2 Valgt løsning

Grunnet de store usikkerhetene knyttet til hvordan en maskinlæringsbasert modell vil kunne fungere, var det nødvendig å begrense denne risikoen. Derfor ble det valgt at både maskinlæring- og algoritme-baserte løsninger skulle testes slik at disse kunne sammenlignes.

Det var problemer i forbindelse med utviklingen av maskinlæringsmodellen, noe som resulterte i at potensialet aldri ble testet. Et av problemene inkluderte spørsmål om hvordan datasettet skulle struktureres på en måte som var kompatibel med de eksisterende testene, samt generell manglende koordinasjon. Derfor ble det valgt en algoritmebasert løsning.

3.3 Valg av verktøy

Oppdragsgiver har ikke stilt krav til bruk av verktøy. En viktig del av startfasen var å utforske hvilke verktøy som kunne brukes til løsning av oppgaven. Her ble det blant annet undersøkt diverse OCR-verktøy for lesing og tolkning av informasjon i notesett. Andre ting som ble diskutert i denne fasen av prosjektet var valg av programmeringsspråk og tjenesteløsning.

3.3.1 Programmeringsspråk

Det er hovedsakelig tre programmeringsspråk som har blitt diskutert. Valg av programmeringsspråk kan påvirke hvilket produkt en ender opp med ettersom de har forskjellige egenskaper.

3.3.1.1 JavaScript

Oppdragsgiver benytter seg av JavaScript i utviklingen av deres løsninger. Ettersom løsningen skal være på tjenestesiden, vil rammeverket Node.js være passende. Node.js er et JavaScript runtime-system som kjøres på tjenestesiden (GeeksforGeeks, 2021). En løsning utviklet i Node.js, ville gjøre det lettere for oppdragsgiveren å videreutvikle løsningen ettersom Styreportalen allerede har kompetanse på dette området. En større

utfordring her er at gruppen har begrenset erfaring med Node.js og JavaScript. Gruppen har dermed lite erfaring med hvilke begrensninger og fordeler det har som verktøy.

3.3.1.2 Java

Java er det programmeringsspråket som gruppen har mest erfaring med. Java er et objektorientert språk og er dermed gunstig i utvikling av blant annet applikasjoner i form av web-applikasjoner, android-applikasjoner og i utvikling av større programmer (JavaTpoint, ingen dato).

3.3.1.3 Python

Python er et programmeringsspråk med stor fleksibilitet og enkel syntaks. Python har en stor brukerbase med lett tilgjengelige biblioteker/verktøy. Flere av dem er godt egnet til prosessering av data. Python brukes også mye i arbeid med maskinlæring. En av hoved ulempene med Python er at det ikke er like raskt sammenlignet med flere andre høynivåprogrammeringsspråk (Prasanna, 2022).

3.3.1.4 Valg

Etter en sammenligning av disse programmeringsspråkene, ble det konkludert med at Python er språket som vil være best egnet i gjennomføringen av prosjektet. Flere av fordelene med Java i forhold til Python, som tid, er ikke særlig relevante. Gruppen har vurdert at usikkerheten med å velge JavaScript med Node.js er større enn den eventuelle nytten det kunne gitt på enkelte punkter.

3.3.2 Tjenerløsning

Tidlig nevnte oppdragsgiver at de benyttet seg av Firebase og Google Cloud Platform (GCP) for sine eksisterende løsninger. Det å benytte den samme løsningen som Styreportalen allerede bruker, vil være en fordel med tanke på at de allerede har kompetanse på området.

GCP består av flere ulike verktøy som virtuelle maskiner og det Google kaller «tjenerløse»-løsninger. En del av disse kunne vært benyttet for å drifte den endelige løsningen (Hummel, ingen dato).

De to løsningene som ble tatt i betraktning var Cloud Functions og Cloud Run. Cloud Functions ble i utgangspunktet valgt. Det er en «Function-as-a-Service» (FaaS) -løsning. Dette er en av Google sine «tjenesteløse»-løsninger. Denne typen løsning gir mulighet for å utvikle enkle funksjoner som er i stand til å levere skalerbare løsninger, uten å måtte forholde seg til tekniske utfordringer knyttet til drift (Google, ingen dato a).

Imidlertid ble det avdekket to ulemper med denne løsningen. Utgangspunktet for prosjektet var at programmet skulle ha ett API-endepunkt, men etter hvert ble det mer komplekst. Dette skapte utfordringer, ettersom Cloud Functions ikke støtter flere

endepunkt. En annen ulempe oppstod i forbindelse med bruken av maskinlæring. Når modellen skulle tas i bruk, var det utfordringer med å rulle ut programmet.

Et alternativ til Cloud Functions er Cloud Run. Denne løsningen gir mulighet til å drifte enkle container-apper. Cloud Run er også skalerbart (Google, ingen dato b). Fordelen med en skalerbar løsning, er at det gir mulighet for å analysere flere notesett samtidig. I dette tilfellet ble Flask-rammeverket brukt for å bygge applikasjonen (Google, ingen dato d).

3.3.3 OCR-verktøy

Det ble også sett på ulike OCR-verktøy. I startfasen ble flere OCR-verktøy som blant annet Nanonets OCR og Adobe Acrobat Pro OCR undersøkt til bruk for oppgaven, men ingen av dem passet helt til det som var ønsket. Det ble konkludert med at Google Vision API var en passende løsning for prosjektet.

Google Vision API fungerer godt med flere populære programmeringsspråk og det oppstod dermed ikke mange problemer med implementeringen av OCR-en i Python. Denne OCR-en har en generelt høy treffsikkerhet i tolkning av utskrevet tekst, slik som de fleste moderne notesett. I tillegg fungerer den også greit for håndskrevet tekst. Med denne OCR-en var det mulig å få ut en JSON-fil med informasjon om teksten på sidene. Dette valget av OCR kan dessuten være en fordel for Styreportalen med hensyn på at de allerede bruker Google Cloud Platform. (Cheng, 2022)

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

Utviklingsmetodikken som blir brukt i prosjektet er en versjon av The Lean Startup metodikken. I «ING303 Systemtenking og Innovasjon for ingeniører» som blir undervist ved HVL for alle ingeniørstudenter, var det en gjennomgang av The Lean Startup. Der er The Lean Startup beskrevet som «en arbeidsmetode for å skape noe folk faktisk vil ha» (Bækkelund, 2022). Det er mye brukt i oppstartsbedrifter, innenfor IT og andre felter som krever utvikling.

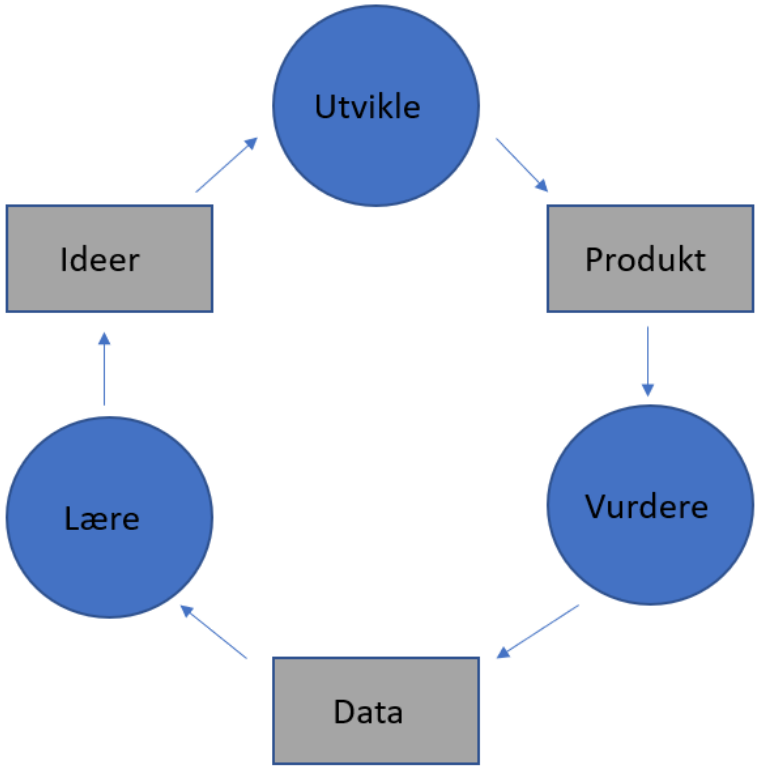
Å bruke en versjon av The Lean Startup, kan hjelpe med å gjøre arbeidet agilt og iterativt. Prosjektet blir tilpasset basert på tilbakemeldinger og vurderinger underveis. Det iterative aspektet gjør at prosjektet deles opp i mindre deler. Dette gjør at en får raskere tilbakemeldinger og bedre oversikt. (Ries, 2019)

The Lean Startup bygger på å lage en såkalt MVP, som står for «Minimum Viable Product». MVP-en kan variere. I oppstartsbedrifter vil MVP bli brukt for å vise fram produktet, få tilbakemelding og finansiering. Ideen er å bruke så liten tid på å finne ut om løsningen du har, faktisk virker og om mulige kunder er villige til å bruke det. (Ries, 2019)

Selv om bachelorprosjekt ikke har samme behov som Startup firmaer når det kommer til finansiering, er ideen bak en MVP nesten det samme. I stedet for penger er det timer som

teller som økonomi. Derfor er det fokus på å bruke kort tid på å finne ut om løsningen for problemstillingen er aktuell. Dette forhindrer at tiden blir brukt på unødvendige aktiviteter. Når en får tidlig tilbakemelding, er det mulig å tidlig vurdere om man er på rett spor eller ikke. Om ting må endres i starten eller dersom prosjektet må begynne helt på nytt, kan det skje etter en uke istedenfor en måned. Når MVP vises til oppdragsgiveren, trenger den ikke å være klar eller funksjonell. Det viktigste er å presentere tilnærmingen. (Ries, 2019)

Lean Startup-metodikken deler hver iterasjon inn i tre faser. Den første iterasjonen er utviklingsfasen, hvor målet er å utvikle en løsning/produkt for å håndtere det identifiserte problemet. Den andre fasen er evalueringsfasen/vurdere. Der gjennomføres det en nøye vurdering av den utviklede løsningen i samarbeid med oppdragsgiveren for å identifisere hva som fungerte og hva som må justeres. Dette gir data og annen informasjon som kan brukes i arbeidet videre. Til slutt kommer lærefasen, der fokuset havner på å lære av både feilene som er gjort og det som har vist seg å fungere. Målet er å oppnå forbedring gjennom refleksjon, tilpasning og få fram nye ideer som legger grunnlaget for å være godt forberedt for neste iterasjon. (Ries, 2019)



Figur 7: Viser flyten til The Lean Startup iterasjon

3.4.2 Prosjektplan

Prosjektet deles i 3 faser. Prosjektplanen var visualisert ved bruk av Gantt-diagram som vises i Prosjekthåndboken (Sterri, Taraldset og Thorsteinsson, 2023b, kap. 1).

Første fasen er en dokumentasjon og forberedelsesfase, der det samles informasjon, litteratur og dokumenter. Disse blir brukt som hjelp for neste fase. Det blir også laget flere dokumenter som visjonsdokument og kravdokument.

Den andre fasen er utviklingen. Utviklingen har blitt delt opp i fire, to-ukers iterasjoner, med mulighet til å legge til to en-ukes iterasjoner ved behov. I tillegg bygger hver iterasjon på den tidligere iterasjonen. Det arrangeres også ukentlige møter med oppdragsgiver for å sikre regelmessig kommunikasjon og oppdatering av prosjektets fremdrift.

Figur 8: Viser iterasjon planen i forhold til uker

Iterasjon 1 og 4 har lengre tid enn 2 uker. Det er fordi at møtene med oppdragsgiver foregår hver torsdag. Dermed starter og slutter iterasjonene på torsdag med unntak av den første iterasjonen. Første iterasjon starter på mandagen i uke 6. Det hadde dermed blitt mindre enn 14 dager, hadde den blitt avsluttet i uke 7. Fordi første iterasjon er viktig, var det bestemt å ha 3 ekstra dager. Iterasjon 4 var i påsken. Det var dermed bestemt at den iterasjonen skulle avsluttes uken etter.

Til slutt i den siste fasen ble det laget en sluttrapport som skal inneholde alt angående prosjektet.

3.4.3 Risikovurdering

En risikovurdering ble utført i starten av prosjektet og ble oppdatert underveis. Ved å vurdere hvilke risikoer som kan oppstå under prosjektets gjennomgang, kan det planlegges i forkant hvordan det er mulig å unngå disse. Risikoene blir vurdert ut ifra sannsynligheten for at det kommer til å skje, multiplisert med konsekvens av at det inntreffer. Der det er gitt karakter på 1-3 på sannsynlighet og konsekvens. Den høyeste risikoen er 9. 1-2 gir grønn farge og har lav risiko. 3-4 gir gul og er en middels risiko. Alt som er 5 eller over får rød farge, har høy risiko og må arbeide aktivt med å unngå. Derfor er det satt opp i en tabell for å få oversikt over risikoen. I tabellen er det satt opp tiltak for å redusere risikoene. I tabell 2 vises, med bruk av risiko matrise oversikt over hvor risikoen ligger i forholdet mellom sannsynlighet og konsekvens

Tabellen i Prosjekthåndbok-vedlegget har flere risikoer som må arbeide mot når det kommer til store prosjektarbeid. Dette inkluderer sykdom, innleveringsfrister og mangel på tekniske ferdigheter (Sterri, Taraldset og Thorsteinsson, 2023b, kap. 2.1).

I dette prosjektet var det 4 spesifikke risikoer. Det handlet om feiltolkning av oppgave, dårlig treffsikkerhet, at API-et brukte for lang tid på å få ut informasjon og at maskinlæringsmodellen ikke gir gode nok resultater til at den kan brukes. I starten av prosjektet var alle disse risikoene høyere.

Etter flere dialoger sammen med oppdragsgiver og bevist arbeid for å unngå risikoene, var det konkludert at konsekvensen av feiltolkning av oppgaven ble trukket ned til middels, men ble holdt på middels sannsynlighet. Dårlig treffsikkerhet har alltid høy konsekvens dersom det er den største delen av oppgaven. Etter mye framgang ble sannsynligheten satt ned til middels. At prosessen tar for lang tid var først høy konsekvens, men etter samtaler med oppdragsgiver var det ikke noe han mente var viktig. Det var viktigere at løsningen var nøyaktig enn at den var rask og derfor ble konsekvensen satt ned på lav. Risikoen rundt det at maskinlæringsmodellen ikke kan brukes, ble ikke endret. Grunnen for det var at det måtte forkastes og dermed klarte ikke gruppen å arbeide mot den risikoen og redusere konsekvensene.

	Hendelse/risiko	Årsak	Sannsynlighet	Konsekvens	Risiko	Tiltak
1	Feiltolkning av oppgaven	Oppgaven blir feiltolket som fører til at ferdige produkt ikke er sånn som oppdragsgiver ville få	2	2	4	Holde god dialog med oppdragsgiver og få tilbakemelding fra han, for hver iterasjon i prosjektet.
2	Dårligere treffsikkerhet på lesing av noter en forventet.	Modellen som brukes, er ikke treffsikker nok sånn som målet er fra problemstillingen.	2	3	6	Arbeide godt med oppgaven. Vær godt i forkant på å lage modellen. Sette opp datoer for testing som svarer derifra kan hjelpe oss med å lage bedre modell.
3	Tar for lang tid å hente PDF fil og sende tilbake JSON filen	Løsningen bruker lengre tid å hente PDF-filen, skrive den til JSON og levere den tilbake en det tar å gjøre det manuelt	3	1	3	Arbeide godt med oppgaven. Vær godt i forkant på å lage løsningen. Sette opp datoer for testing som svarer derifra kan hjelpe oss med å lage bedre modell.
4	Maskin-lærings-modell ikke klarer å brukes.	Maskinlærings-modellen klarer ikke å løse det den krever	3	3	9	Være bevist på at dette er noe som krever tid, arbeide godt med det. Dersom den ser som ikke er mulighet. Dropp den.

Tabell 2: Risiko tabell. Inneholder hendelse/risikoen, årsaken, sannsynligheten, konsekvensen, risikoen og tiltakene for å unngå disse

3		4
	1	2

Tabell 3: Risiko matrise. 3 i bredden og 3 i høyden. Sannsynlighet er på høyden og konsekvens i bredde retning

3.5 Evalueringsplan

En viktig del av et utviklingsprosjekt er evalueringen. På slutten av hver iterasjon er det en evaluering for å se om det foreløpige produktet svarer på problemstillingen. Her vil oppdragsgiver være involvert for å se om løsningen returnerer informasjon på en forventet måte. Andre mangler vil også kunne plukkes opp under disse evalueringene.

I dette prosjektet lages det en løsning som skal hente ut informasjon fra en PDF-fil. Det er urealistisk at løsningen vil kunne få korrekt resultat i alle tilfeller, ettersom unormale oppsett og dårlig kvalitet kan gjøre PDF-filene vanskelige å tolke. En viktig del av evalueringen blir dermed å vurdere treffsikkerheten av løsningen. Ved slutten av hver iterasjon, var det derfor viktig å finne ut hvor godt løsningen fungerte, og hvilke tilfeller den slet med.

Ved den endelige evalueringen, vil det være lurt å gjøre en grundigere gjennomgang for å finne ut nøyaktigheten til løsningen. Det er viktig at dataen den testes på er representativ i forhold til dataen løsningen skal brukes på. Dette kan bli en større feilkilde.

Som en del av evalueringen er det viktig at det blir evaluert på en datamengde som ikke har blitt brukt under utviklingen. Dette har med at programmet som har blitt utviklet vil være tilpasset de notesettene som er brukt under utviklingen, og risikerer dermed å gjøre det bedre på disse enn den gjør generelt.

3.5.1 Verktøy

Som en del av evalueringen er det nyttig å ha verktøy som kan gjøre denne prosessen raskere. En av utfordringene med evalueringen, er at dataen ikke har navngitt hva som er det ønskede svaret på forhånd. Derfor trengs det et verktøy som kan brukes for å gjøre evalueringen raskere. Utviklingen av dette har vært en del av prosjektet.

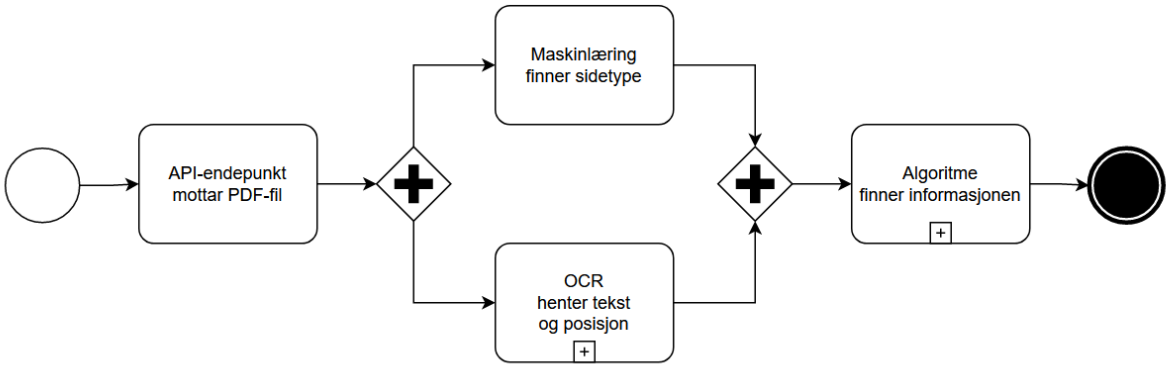
En av grunnene til at dette kan være nyttig, er relatert til at oppdragsgiveren uttrykte muligheten for at de skal kunne utvide løsningen. For eksempel kan den utvides til å akseptere flere språk. Da kan det være nyttig med et verktøy som gjør det enkelt å vurdere om løsningen fungerer som forventet.

4 Detaljert løsning

Hovedmålet med programmet som er utviklet, er å hente ut informasjon på et notesett. Det er også den største delen av løsningen. Uthentingen kan beskrives i en flyt. Flyten starter med et API-endepunkt som mottar en PDF-fil.

Etter dette startes to prosesser som brukes for å hente ut data om PDF-filene, som teksten og «sidetypene». Maskinlæring og OCR-prosessene kjøres samtidig (concurrent). Dette gjøres ved hjelp av Python sine innebygde tråder. Python-trådene kjører ikke parallelt (på forskjellige CPU-kjerner), men oppnår fremdeles en økt hastighet i dette tilfellet. Dette har med at OCR-modulen er I/O-begrenset (Python Software Foundation, ingen dato).

Når disse har fullført, sendes denne informasjonen til en algoritme som brukes for å hente stemmene, tittelen, komponisten og arrangøren. Dette returneres så i form av en JSON-fil.



Figur 9: BPMN-flyt som beskriver den overordnede prosessen (BPMN 2.0 Symbols - A complete guide with examples., ingen dato)

Denne delen av rapporten er bygget opp etter flyten.

4.1 API-endepunkt

En sentral del av prosjektet er knyttet til det å lage et hensiktsmessig API-endepunkt. API-et skulle i utgangspunktet kun omfatte en ting; motta en PDF-fil og returnere informasjonen om notesettet.

Applikasjonen er laget ved hjelp av Flask-rammeverket. Flask er et Python-rammeverk som kan benyttes til å lage enkle nettløsninger. I dette tilfellet benyttes det til å lage et API-endepunkt (Pallets, ingen dato).

API-endepunktet er satt opp til å kunne motta et POST-request. Her sjekkes filtypen ved å kontrollere at content-typen-headeren er satt til «application/pdf». Hvis endepunktet mottar noe som ikke passer med dette, sendes det en feilmelding i retur. Hvis noe skulle gå galt under kjøringen, sendes det også en feilmelding.

Etter hvert kom det også et ønske fra oppdragsgiver om at det skulle være mulig å oppdatere en instrumentliste. Denne listen brukes som en del av algoritmen. Her kan det kjøres et kall for å oppdatere instrumentlisten ved å slette eller legge til elementer.

En nærmere beskrivelse av API-endepunktet, og hvordan de brukes finnes i Systemdokumentasjonen (Sterri, Taraldset og Thorsteinsson, 2023c, kap. 6).

4.2 Maskinlæringsmodell

Behovet for å kunne klassifisere sider ble tydelig under utviklingen av algoritmen. For eksempel kan det å vite at en side er partitur være nyttig. Måten algoritmen finner «stemmen» på, vil i flere tilfeller returnere en stemme istedenfor å identifisere det som en partiturside. Partitursider inneholder ofte en liste med alle stemmene i notesettet. Når algoritmen finner disse stemmene i partiturlisten, brukte den i en tidlig fase av utviklingen den første stemmen den fant og returnerte denne som stemmen på siden. Dette skal den ikke gjøre. Metodene til den forrige gruppen for å identifisere partitursider ble testet. Metoden var å telle antall stemmer den finner på siden. Oversteg den en grense, ble siden en partitur-side. Her oppstod det et problem i den nye løsningen ved at stemmene ofte er forkortet. I flere tilfeller ble «Oboe» returnert som stemme ettersom dette er den eneste som ikke ble forkortet og de forkortede stemmene ble ikke funnet.

Derfor var det behov for å finne en måte å skille mellom disse. Det er også andre sidetyper som er relevante for algoritmen. Sidetypene predikeres er:

Sidetype	Beskrivelse
Første note	Første side til en stemme. Disse har som regel informasjon som ikke finnes på de neste sidene, dersom stemmen går over flere sider.
Note	Sider som ikke passer inn under kategoriene over. Altså sider som hører til stemmer, men ikke den første siden.
Partitur	Et partitur i notesett er en sammenstilling av alle stemmene sine noter i et musikkverk. Det viser alle stemmene, både vokale og instrumentale, og deres respektive melodier og tekster, på et enkelt noteark (Nagelhus, 2008). Disse sidene ser forskjellige ut fra andre notesider.
Annet	Dette er sider som ikke er notesider. Altså forsider, biografier, blanke sider osv.

Tabell 4: Oversikt og forklaring av de ulike sidetypene som benyttes.

For å skille mellom ulike «sidetyper» blir det brukt en maskinlæringsmodell.

4.2.1 Henting av data

Den største utfordringen med å trene en maskinlæringsmodell er å konstruere et godt datasett som modellen skal trenes på. Denne vurderingen tar hensyn til flere faktorer, slik som størrelsen på datasettet og om datasettet er variert og representativt. Uten et representativt datasett, vil det ikke være mulig å få et godt resultat uavhengig av modellen. Dersom det ikke blir tatt hensyn til disse faktorene, vil nøyaktigheten blir dårlig (Géron, 2019, kap. 1).

Dette var en av de utfordringene som ble diskutert under utviklingen. En rekke forskjellige notesett ble innhentet fra Styreportalen og benyttet i konstruksjon av datasettet til maskinlæringsmodellen. Modellen sin nøyaktighet avhenger av at datasettet fra oppdragsgiver er variert og representativt.

For å lage dette datasettet som modellen skal trene på, ble det laget et skript som lagrer alle sidene i PDF-filene i datasettet som bilder. Skripet bruker data fra en manuell gjennomgang av PDF-filene, for å bestemme hvilken kategori hvert bilde hører til. I den manuelle gjennomgangen ble navnet på stemmen for hver side lagret. Om det ikke er en partiturside eller «annet», kan konteksten fra sidene rundt brukes for å finne ut om det er første siden til en stemme eller ikke. Bildefilene lagres i forskjellige mapper alt etter kategorien som brukes. Filene fikk navnet fra den originale PDF-filen pluss sidetall som filnavn.

Denne prosessen gir ikke alltid ønsket resultat. Dette kan ha med feil i den manuelle gjennomgangen. Hovedproblemet er at i enkelte notesett er stemmene gjentatt flere ganger. Det blir da flere sider som hører til kategorien første side av en stemme, plassert i mappen med sider som ikke er førstesiden. Fastai har verktøy som kan benyttes for å finne data som er kategorisert feil (fastai, ingen dato b).

Et problem som kan oppstå under trening av en modell, er at den blir overtilpasset. Det vil si at den lærer å kjenne igjen dataen den trener på fremfor å «lære» generelle regler. I praksis betyr dette at en modell som blir overtilpasset datasettet, ikke kommer til å klare å predikere på ny data. For å unngå dette problemet, er det viktig å ha en metode for å kunne teste nøyaktigheten av modellen. Dette gjøres ved å teste modellen på data som den ikke ble trent på. Datasettet blir derfor delt opp i treningsdata og valideringsdata (Howard, Gugger og Chintala, 2020, kap. 1). Som regel kan datasettet deles opp tilfeldig. I tilfellet med sider av notesett, kan det skape problemer fordi modellen lærer å gjenkjenne notesettet. Den lærer hvordan en partiturside ser ut for hvert av notesettene i dataen, og ikke hvordan en partiturside ser ut generelt. For å unngå dette, er det laget en metode som splitter datasettet på en slik måte at overlappene notesett blir minst mulig. Dette blir gjort ved å sortere arkene alfabetisk og dele av den nederste 30 prosenten som ble brukt som valideringssett.

Et annet tiltak som er brukt for å hindre overtilpassing, er «data augmentation». Dette er et verktøy som brukes til å gjøre datasettet «større» ved å modifisere og lage tilfeldige variasjoner av bildene i datasettet uten å endre innholdet i bildene. I dette tilfellet er det blant annet brukt rotasjon og endring av lysstyrke (Howard, Gugger og Chintala, 2020, kap. 2).

4.2.2 Trening av modellen

For trening av modellen, ble det valgt å benytte ResNet som er en forhåndstrent dyplæremodell. Det finnes flere varianter av denne modellen som blant annet ResNet18, ResNet34 og ResNet50. Forskjellene mellom disse variantene er antall lag og parametere i modellene. ResNet34 har flere lag enn ResNet18. Denne økningen i kompleksitet gjør ResNet34 mer utsatt for overtilpassing, men er samtidig mer nøyaktig enn ResNet18. ResNet18 derimot har færre lag og parametere, noe som gjør den raskere enn ResNet34 (Howard, Gugger og Chintala, 2020, kap. 5).

Til tross for at ResNet34 i utgangspunktet er tregere enn ResNet18, er ikke det betydelig med hensyn på at modellen blir kjørt samtidig som OCR-en. Begge modellene bruker mindre tid enn OCR-en. Dette gjør ResNet34 til et attraktivt valg, da den tilbyr høyere nøyaktighet uten at det går på bekostning av hastigheten.

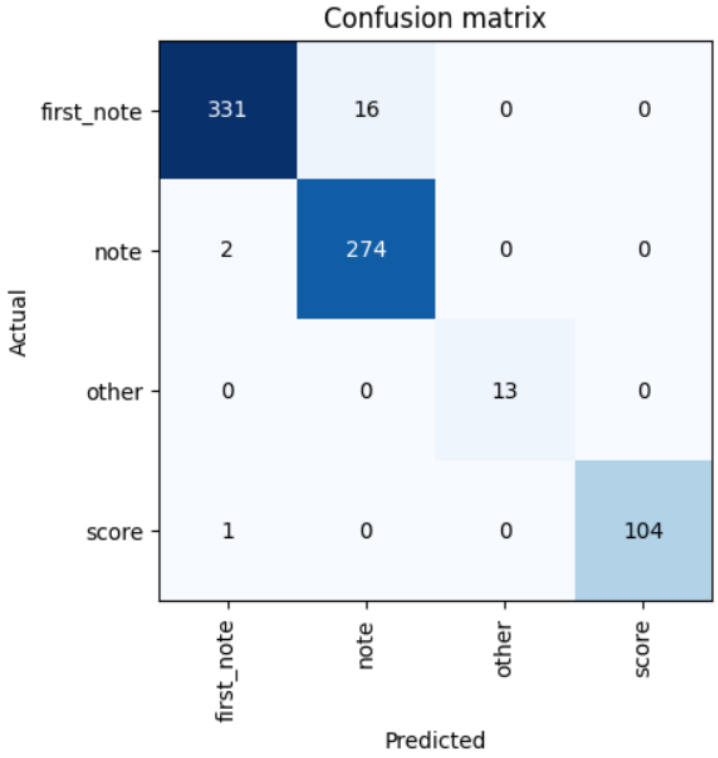
Modellen ble trent på 11 epoker. En epoke refererer til trening av modellen på treningssettet en hel gang. I dette tilfelle betyr det at modellen trenes på treningssettet 11 ganger. En konsekvens av å trene modellen på flere epoker, er at på slutten av hver epoke memorerer modellen mer av treningssettet. Her må det passes på at den ikke blir trent på for mange epoker med tanke på at det kan føre til overtilpassing på treningssettet. På samme måte dersom modellen blir trent på for få epoker, blir ikke modellen like nøyaktig slik som med flere epoker (Howard, Gugger og Chintala, 2020, kap. 5).

epoch	train_loss	valid_loss	error_rate	time
0	0.234190	0.290005	0.080972	02:56
1	0.133865	0.173088	0.051282	02:51
2	0.093183	0.249695	0.068826	02:39
3	0.124079	0.304635	0.076923	02:39
4	0.073112	0.203141	0.047233	02:40
5	0.040793	0.140305	0.036437	03:17
6	0.024035	0.087857	0.025641	04:12
7	0.010145	0.103220	0.031039	04:27
8	0.005830	0.109258	0.028340	04:12
9	0.007908	0.099492	0.025641	04:11
10	0.007200	0.131015	0.025641	04:15

Figur 10: Utskift fra «fine tuning» av modellen over 11 epoker.

Tabellen viser resultatene fra trening av modellen etter 11 epoker. Den kolonnen som er mest interessant, er «error_rate» ettersom den beskriver nøyaktigheten av modellen. Det er ønskelig at denne verdien skal være så lav som mulig. Det er tydelig at nøyaktigheten forbedres etter hver epoke da feilprosenten som regel synker. Andel feiltolkninger ligger på 2,6% etter 11 epoker.

Selv om feilprosenten er relativt lav, predikerer modellen fortsatt feil på noen områder. Da vil det være naturlig å visualisere en «confusion-matrix» som kan gi en ide om hvor modellen predikerer feil (Géron, 2019, kap. 3).



Figur 11: Confusion-matrix som viser hvilke feil modellen gjør.

Ved hjelp av en slik confusion-matrix, er det mulig å se hvilke feil den gjør. Ut ifra observasjoner ser gruppen at de fleste feilene er gjort der noten skal være klassifisert som «first_note», men er predikert «note». Dette er forventet med hensyn på at første noteark er forholdsmessig lik alle andre noteark og det kan derfor være utfordrende for modellen å skille kategoriene.

4.2.3 Implementasjon

Etter at en modell er laget, kan den eksporteres til senere bruk. Fastai-biblioteket har verktøy for å lage modellen som en pickle-fil. Ved oppstart av programmet, vil modellen bli ekstrahert fra pickle-filen (fastai, ingen dato a). For å kunne predikere på en side er det nødvendig å konvertere sidene i PDF-en til bilder. Dette gjøres ved hjelp av pypdfium2, et Python-bibliotek som kan brukes til å behandle PDF-filer (pypdfium2-team, ingen dato). Hver side kjøres gjennom modellen, og de predikerte verdiene sendes videre til algoritmen.

4.3 OCR-en

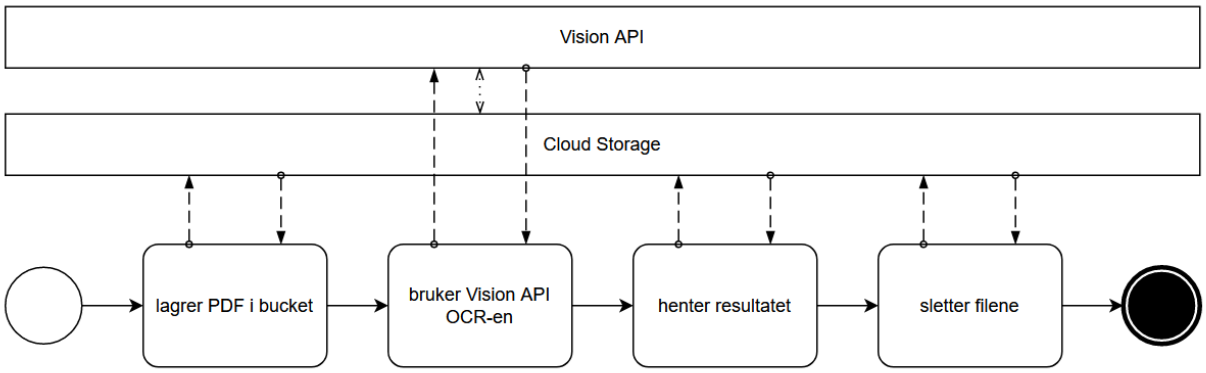
OCR-modulen består av en kontroller som kaller på ulike manager-objekter som har kontakt med API-ene som benyttes i løsningen.

4.3.1 Vision API

Vision API blir benyttet som OCR. For å bruke OCR-en på en PDF-fil, må filen lastes opp til en Storage Bucket. Når det gjøres et kall til API-et, må destinasjonen til input og output oppgis. Det betyr at for å bruke OCR-en i dette tilfellet, er en nødt til å benytte Cloud Storage som et mellomledd (Google, ingen dato c).

4.3.2 Cloud Storage

Som en del av prosessen rundt å bruke Vision API, kreves det at PDF-filen lagres i en Storage Bucket. Resultatet fra OCR-en lages også her. Programmet henter resultatet fra OCR-en. Ettersom det ikke er noe behov for å oppbevare PDF-en eller resultatet i Bucketen, blir disse slettet før resultatet sendes videre til algoritmen.



Figur 12: BPMN-modell som beskriver subprosessen. Modellen viser til meldinger som sendes mellom programmet og Vision API og Cloud Storage. Det er også markert at disse kommuniserer med hverandre. (BPMN 2.0 Symbols - A complete guide with examples., ingen dato)

For å forhindre konflikt om flere filer lastes opp samtidig, generes det unike navn som brukes på input og output-filen. Disse baseres på tid og en hash-verdi av filen. Alle filene plasseres under «t/[unikt navn]/». I tilfellet det skjer en feil som hindrer at filene slettes, er det lagt inn regler som sørger for at filer under mappa «t/» som er eldre enn en dag, slettes.

Grunnen til at det er nødvendig å legge filene under «t/» er at det også lages en instrumentliste-fil, som er nødvendig i algoritmen. Denne skal ikke slettes og er den eneste filen i bucketen som ikke plasseres i «t/».

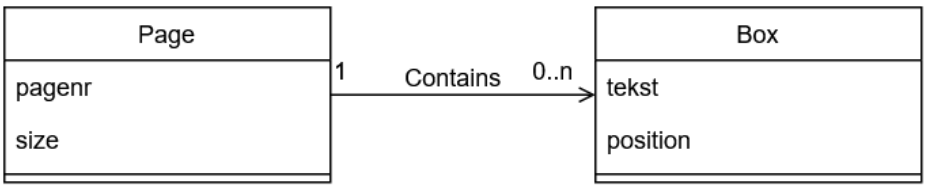
4.3.3 Behandling av OCR-data

OCR-en returnerer en JSON-fil. Filen som hentes ut inneholder teksten delt inn i tre forskjellige nivåer med «bokser». Nivåene er «blocks», «paragraph» og «word». Hver av

disse har forskjellig informasjon som posisjon, språk og usikkerhet. Det er behov for å prosessere dataene for å få den på et format som kan benyttes.

I denne prosessen fjernes noe av informasjonen. Blant annet usikkerheten knyttet til teksten. Usikkerheten kunne blitt brukt til å flagge plasser hvor en kan drive med etterprosessering for å korrigere feil. Ettersom OCR-en ser ut til å returnere riktig tekst i de fleste tilfeller, er det ikke stor gevinst av å benytte dette. Et annet scenario som har blitt oppdaget, er at i flere tilfeller returnerer OCR-en tekst som ikke eksisterer som f.eks. «போன் ©ttpPandன்மன்/». Det kunne vært mulig å benytte flaggingen her. Det å sørge for at kun teksten øverst på arket blir sett på, fjerner de fleste tilfeller av dette. Denne type etterprosessering kan også skje før informasjonen sendes til algoritmen. Det er dermed ikke behov for at algoritmen får tilgang til denne dataen.

En grunn til å ikke sende all informasjon, er å gjøre programmet mer fleksibelt. Det å standardisere formatet gjør det lettere å erstatte Vision API, med andre verktøy, uten å måtte gjøre endringer på algoritmen. I dette tilfellet mottar algoritmen informasjonen i form av et objekt. Objektet skal være av en klasse som implementerer to metoder. Begge metodene returnerer informasjon fra sidene, i form av Page- og Box-objekter, men med forskjellige regler for hva en boks er.



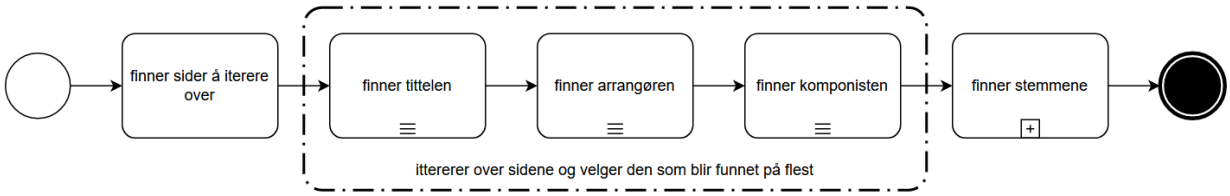
Figur 13: Demonstrerer forholdet mellom Page- og Box-objekter.

I tilfelle med Vision API vil `get_boxes`-metoden gi tilbake informasjonen basert på boksene på «paragraphs»-nivået til Vision API. Teksten og koordinatene til hjørnene på boksen lagres i Box-objekter. Disse plasseres i en liste i Page-objektene. Page-objektet inneholder sidetall og størrelsen på sidene. `get_boxes`-metoden returnerer en liste med Page-objekter.

Den andre metoden er `get_single_line_boxes`. Den returnerer bokser som bare inneholder en linje. Klassen som implementerer dette, lagrer JSON-resultatet fra OCR-en. Hvis de ikke allerede eksisterer, vil den lage listene med Page-objekter, og returnere dem.

4.4 Algoritme

Verdiene som kommer ut fra OCR-en og maskinlæringsmodellen, benyttes videre i programmet. Denne delen er algoritmen som bestemmer det endelige resultatet av programmet. Algoritmen skal finne fire ting. Stemmen, komponist, arrangør og tittel. Ettersom komponist, arrangør og tittel er punkter som ofte gjentas på flere sider, behandles disse på en annen måte enn stemmen.



Figur 14: Demonstrerer subprosessen for algoritmedelen. (BPMN 2.0 Symbols - A complete guide with examples., ingen dato)

Et problem med OCR-resultatet, er at Vision API sin standard gruppering, ofte plasserer komponisten og arrangøren i samme boks. Dette gjør det vanskelig å skille mellom disse to. Derfor benyttes det enkeltlinjeboksene (`get_single_line_boxes`). Enkeltlinjeboksene brukes også for tittelen.

Selv om denne informasjonen gjentas flere ganger gjennom notesettet, er det ikke alle sider som har den. Basert på notesettene fra oppdragsgiveren, har den første siden av en stemme (her referert til som førsteside) som regel tittel, komponist og arrangør på «standardiserte» oppsett. Algoritmen som er utviklet baserer seg på dette oppsettet.

For å hente ut de relevante sidene, benyttes sidetypene fra maskinlæringsmodellen. Det lages en liste over alle sidene som er førsteside. Hvis det ikke finnes en førsteside, som PDF-er som bare består av partituret, velges den første partitursiden. Denne har ofte et lignende oppsett som en normal førsteside med tanke på tittel, komponist og arrangør.

Algoritmene som brukes for å finne tittel, komponist og arrangør, kjøres for alle sidene som er funnet. Løsningen returnerer det resultatet som ble funnet flest ganger. Dette kommer av at det kan være små forskjeller mellom sidene som gjør at kjøringen bør gjentas på flere sider. Et eksempel på dette er at tittelboksen kan inneholde stemmen i tilfeller der de blir for nærme.

4.4.1 Tittel

Tittelen finnes med å gå gjennom listen av bokser på siden. For å være en kandidat, må boksen være på den øverste delen av siden. Blant disse boksene sammenlignes høyden. Ettersom enkeltlinjeboksene blir benyttet, vil høydene kunne brukes til å si noe om størrelsen på fonten.

Et problem som kommer med dette, er at det finnes i enkelte tilfeller vertikale bokser. For å forhindre at disse velges, filtreres også de boksene der høyden er større enn bredden.

4.4.2 Arrangør

Basert på eksempelnotesettene, ble det oppdaget at boksen som inneholder navnet på arrangøren, startet i de fleste tilfellene med enten «Arr», «Arrangør» eller tilsvarende. Dermed var det mulig å benytte dette for å finne den riktige boksen. Boksen er som regel plassert på høyre side.

For å finne arrangøren, brukes den første boksen på høyre side som starter med «arr». Forskjellen på store og små bokstaver blir ignorert. Det blir også gjort en enkel etterprosessering av det som er funnet. I utgangspunktet skal det returnere «Jerker Johansson», men boksen inneholder «Arranged by Jerker Johansson». Det første ordet fjernes og hvis det andre er «by» eller «av», fjernes det også.

Disse reglene vil som regel finne arrangøren, men det finnes flere unntak av dette, som at det f.eks. står «edited by». Dette blir til en viss grad tatt hånd om ettersom løsningen også aksepterer det som en arrangør om boksen inneholder «edited», men det demonstrerer et problem med løsningen. Løsningen er avhengig av hardkodede regler. Hvis det står «transcribed by», «inst.», eller en versjon av disse, klarer ikke løsningen å finne arrangøren. Det er ikke en generell løsning, men en som er tilpasset akkurat eksemplene som har blitt brukt til å lage løsningen.

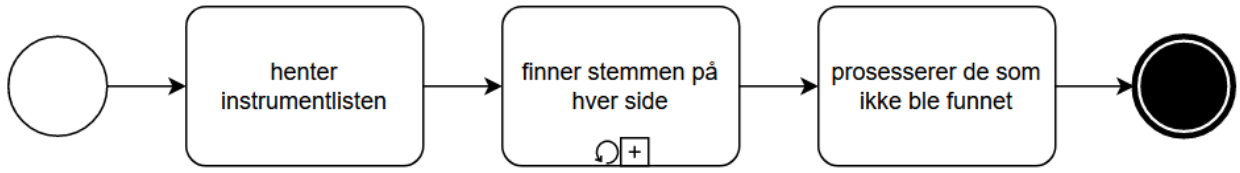
Figur 15: Viser stemme til venstre, tittel i midten, komponist øverst til høyre og arrangør den nedre til høyre (Evart og Jerker Johansson, ingen dato)

4.4.3 Komponist

For å identifisere komponister, benyttes arrangøren som kontekst. Det er boksene øverst på høyre side som blir vurdert og det er den høyeste boksen som blir valgt. Dette er begrunnet i eksemplene der det sjeldent finnes tekst på høyre side som overstiger komponistinformasjonen.

Det oppstår i imidlertid fortsatt utfordringer ved identifisering av komponister der de ikke finnes. I slike tilfeller blir arrangøren satt som komponist. For å forhindre dette, blir arrangørboksen ignorert. Hvis det står tekst under arrangøren, kan dette føre til at algoritmen anser dette som komponisten. Derfor blir løsningen begrenset av kravet om at dersom det finnes en arrangør, må komponisten være plassert over arrangøren.

4.4.4 Stemmen

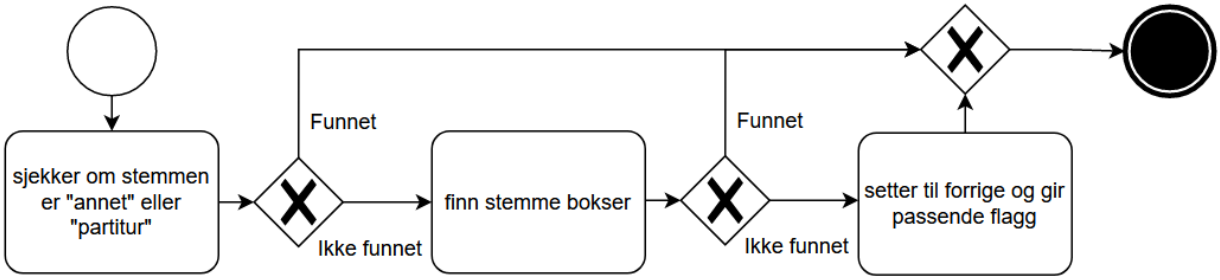


Figur 16: BPMN-modell som beskriver hvordan stemmen er funnet. (BPMN 2.0 Symbols - A complete guide with examples., ingen dato)

Algoritmen som brukes for å finne stemmen, er mer kompleks. Den krever listen over alle sider og listen av sidetypene. I dette tilfellet brukes Vision API sine bokser, ikke enkeltlinjeboksene som i de andre. Bakgrunnen for dette er at stemmen i enkelte tilfeller kan gå over flere linjer. Det trenger også tittelen. Dette har med at tittelen og stemmen, av og til havner i samme boks. Her trengs en liste over instrumenter som hentes fra Google Storage.

For å finne stemmen på en side, finnes det tre faser.

—



Figur 17: Subprosess som beskriver hvordan stemmen er funnet på en enkel side. (BPMN 2.0 Symbols - A complete guide with examples., ingen dato)

4.4.4.1 Annet eller partitur

Den første delen bestemmer om det er en «annet»- eller «partitur»-side. Er sidetypen «annet», settes den til «annet» uten noe videre vurdering. Noe av grunnen til dette, er at maskinlæringsmodellen ser ut til å være flink til å skille ut «annet»-sider. I tillegg, oppstår det problemer dersom avgjørelsen blir tatt senere. Ofte finnes det navn på stemmer eller instrumenter på «annet»-sider.

Det blir lagt større vekt på usikkerheten knyttet til partitur. Ettersom sidetype ikke er helt sikkert, brukes konteksten fra sidene rundt for å bestemme om det er en partiturside. Hvis ingen av nabosidene er en partiturside, blir denne ikke satt til partitur. Om begge nabosidene er partitur, settes den til partitur. Dette er for å hindre at enkeltsider som er feilkategorisert, gir feil utslag.

Hvis det ikke skjer noen konklusjon her, altså at det ikke er annet eller partitur, går algoritmen videre til neste fase.

4.4.4.2 Finn «stemmeboksen»

Stemmen kan gå over flere linjer. Dette er et problem, ettersom viktig informasjon kan mistes. Derfor bruker algoritmen alle boksene som oppfyller kravene. De mulige stemme-boksene begrenses til de som er i øvre venstre hjørne. For å finne en «gyldig» boks, finnes det en instrumentliste som inneholder tekst som er normalt å finne i stemmen.

Dette er en videreutvikling fra det tidligere bachelorprosjektet. Der ble det brukt en database. Hvis det var en boks som lignet på en tekst i databasen, ble dette brukt som stemme. Ettersom det ikke skal returneres en standardisert stemme, kan dette forenkles. Listen til den tidligere løsningen inneholdt hele navnet på stemmen, altså «1 Bb Trumpet». Det er mange varianter å skrive dette på. Dette kan bli en lang liste av synonymer. Om det ikke er snakk om en standardisert liste, så trengs kun instrumentet. Det er altså mulig å se om boksen har en delstreng som finnes i instrumentlisten.

3rd cornet
3rd bb cornet
cornet 3
cornet iii
3. kornett
kornett 3
kornett iii

Tabell 5: Tabell som demonstrerer ulike synonymer for den samme stemmen.

Om et element i instrumentlisten finnes i teksten er det en gyldig kandidat. Det gjøres noe enkel prosessering av teksten slik at tittelen fjernes sammen med ordet «special». Til slutt settes listen over kandidater sammen til en tekst. Dette gjøres for å ikke miste informasjon i tilfelle stemmen skulle havne i forskjellige bokser. Om den ikke finner en stemme går den over til neste steg.

4.4.4.3 Ingen stemme funnet

Ble det ikke funnet en stemme, er det ofte fordi det ikke er en første side til en stemme. I slike tilfeller, finnes ofte ikke stemmen øverst i venstre hjørne.

Derfor brukes sidetypen for å indikere om det er en ny stemme eller om siden hører til den forrige. Om sidetypen tilsier at det ikke er førsteside av en stemme, settes stemmen til det samme som den forrige. Her settes det også et «flagg» som returneres sammen med stemmen i den endelige JSON-filen som sier at det er forventet at stemmen er det samme som den forrige.

Er det ikke den forventede stemmen, settes et annet flagg. I tillegg lagres denne stemmen i en liste som kan brukes senere for å korrigere feil.

—

Etter at alle sidene har fått tildelt en stemme, går algoritmen gjennom listen over sider som er feil. Konteksten brukes fra sider der stemmen er funnet. Gjennom testing, observerer vi at boksen der stemmen står, ofte er på samme plass i bokslisten på de forskjellige sidene. Den mest normale posisjonen blir valgt som den nye boksen og korrigerer de etterfølgende sidene om nødvendig.

Dette hjelper med å finne den rette stemmen selv om de ikke finnes i listen. Det ble gjennomført et eksperiment for å teste effekten av det. I en normal måling, med en fullstendig liste, klarte løsningen å finne riktig stemme i 98% av tilfellene. Etter å ha laget en ny tilfeldig liste med halvparten av instrumentene falt nøyaktigheten til 93%. Dette viser at den klarer å fange opp flere av stemmene som ikke har et instrument i listen.

Den bestemte informasjonen returneres til brukeren.

4.5 JSON-filen

JSON-filen som returneres er bygd opp på en hensiktsmessig måte. Tittel, komponist og arrangør inkluderes. I stedet for at sidene organiseres i en liste, brukes heller «navn» på hver av sidene som tilsvarer sidetallet. Dette er etter ønske fra oppdragsgiver. Oppdragsgiveren ønsket at sidetallet skulle være inkludert. Som en del av dette inkluderes også antallet sider i JSON-filen for å gjøre det lettere å jobbe med sidene.

```
{
    "pages": {
            "1": {
                "part": "[navnet på stemmen]"
            },
            "2": {
                "part": "[navnet på stemmen]",
                "flag": "[informasjon om den uthentede stemmen]"
            },
            ...
        },
    "title": "[tittelen]",
    "composer": "[komponisten]",
    "arranger": "[arrangøren]",
    "number_of_pages": [totalt antall sider]
}
```

Figur 18: Mal for JSON-filen som returneres av programmet

For hver side finnes det en «part». I denne plasseres stemmen. Hvis stemmen er en partiturside eller «annet», settes «part» til «score» eller «other». Det blir lagt til et «flagg» som gir informasjon om hvordan stemmen ble funnet.

Flagg	Betydning
Expected same as previous	Den har ikke funnet stemmen, men forventer at det ikke er den første siden av en stemme, så den velger stemmen til den forrige siden.
Position based part	Den fant ikke et instrument fra instrumentlisten, men forventet at det skulle være en ny stemme. Finner stemmen basert på posisjonen til andre stemmer.
Position based, previous part	Samme som «Expected same as previous», bare at den originale stemmen er posisjonsbasert.
Expected new part	Stor usikkerhet. Forventer en ny stemme, men klarer ikke å finne den. Velger det samme som forrige stemme.
Unknown first page	Stor usikkerhet. Hvis den ikke finner stemmen, og ikke tror det er «partitur» eller «annet»-side, setter stemmen til «other» og får dette flagget.
Unknown previous part	Stor usikkerhet. Hvis den forventer at det ikke er en ny stemme og forrige side er «Expected new part», «Unknown first page» eller «Unknown previous part», settes dette.

Tabell 6: Tabell for å vise flagg og betydningene

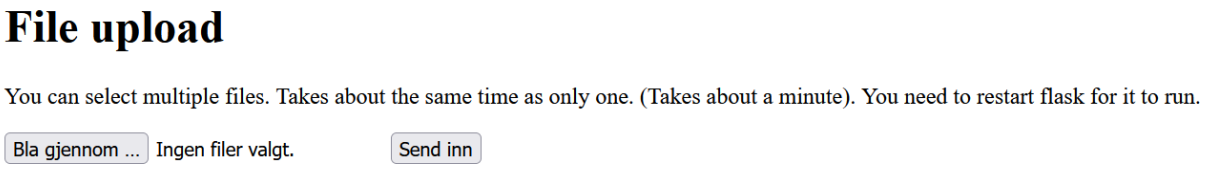
4.6 Testing

Som en del av prosjektet, ble det også laget et system som kunne brukes i testingen. Dette brukes for å gi sidene navn, altså hva løsningen skal returnere. Evalueringen ble dermed lettere underveis i utviklingen.

Testprogrammet kan deles inn i to deler. En for opplasting og navngiving.

4.6.1 Opplasting

Som en del av testingen, er det behov for å lagre filene til Cloud Storage. Etter dette kjøres det kall til Vision API for å få de uprosesserte JSON-filene som returens fra OCR-en. Disse filene brukes under testingen. I testingen kjøres ikke PDF-ene gjennom Vision API igjen. I stedet brukes JSON-filene som ble lagret under opplastingen. Dette er for å spare tid, men også ressurser.



Figur 19: Viser opplastningssiden

4.6.2 Navngiving

En viktig del av testingen er å ha navn som kan benyttes under testingen. Derfor er det laget en side som kan brukes til å gi navn på de ulike PDF-filene.

Labelingprogrammet har flere mangler som blant annet at programmet må starte på nytt for å få med seg de nye filene. Ettersom dette ikke er en direkte del av løsningen, har disse tingene ikke blitt prioritert.

Etter at programmet er startet, vil det ta litt tid før den første PDF-en er klar. Så lenge det finnes nye umarkerte PDF-filer, vil «get next»-knappen gi en ny PDF.

Figur 20: Viser hvordan manuell testing var laget (Meyerbeer, Ptikin og Trevitz, ingen dato)

På den nye siden finnes «Forrige side»- og «Neste side»-knapper brukes til å navigere mellom sidene.

«Annet» og «Partitur» setter stemmen til den gitte siden til annet (forsider, tekst, osv.) eller partitur. «Forrige» setter stemmen til verdien den forrige siden ble satt til. Tekstfeltet brukes til å skrive inn stemmen. Applikasjonen kommer med en liste med forslag til tekst. Denne teksten finnes på siden og er gruppert i boksene som brukes i modellen.

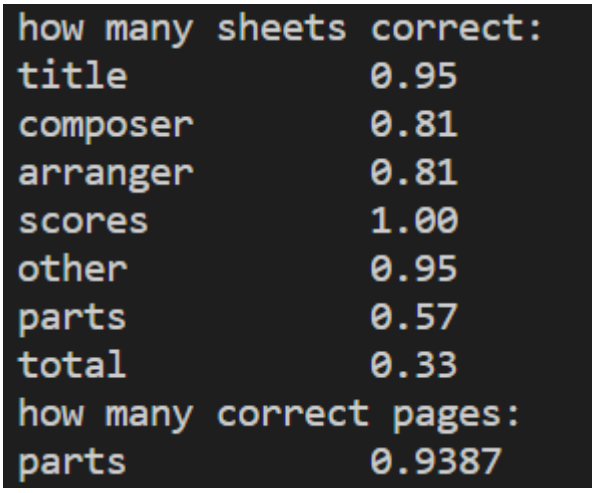
Tekstfeltene under brukes til å sette de andre verdiene.

Når siste side har fått et label, vil det dukke opp en knapp som heter «Ferdig» som lagrer verdiene.

4.6.3 Testing

For å teste programmet, kjørers to Python-filer. Den første går gjennom de ulike filene i Cloud Storage. For hver label-fil lastes den tilsvarende PDF-filen som kjøres gjennom maskinlærings-modulen, og OCR-resultatet gjøres til et format som algoritmen kan akseptere. Dette sendes inn i algoritmen og resultatet fra algoritmen lagres sammen med de ønskede navnene.

Disse blir senere sammenlignet med hjelp av den andre Python-filen. Dette verktøyet gir «scorer» på hvor bra løsningen har gjort det.



Figur 21: Viser prosent av riktig lesing

Scoringen er basert på antall sett som ble riktig. Hva som regnes som riktig kan variere fra oppsett til oppsett. Kjøringen er laget på en slik måte at vurderingene for hva som regnes som like tekststrenger, lett kan byttes ut.

Under testingen, har det vært mer hensiktsmessig å lete etter riktig «boks» i stedet for riktig tekst. Grunnen til dette er at teksten ikke alltid stemmer helt med det som er funnet. Dette kan være feil med OCR-en eller andre feil som at den får med seg ekstra tekst som kan være utfordrende. Derfor er det dette som har blitt brukt videre.

Det som har blitt brukt er Jaro-Winkler-distansen. Den tidligere bachelorgruppen hadde erfaring med å bruke denne. Der ble det funnet ut at dette fungerte bra for å finne stemmen. Jaro-Winkler-distansen er en måte å måle forskjellen mellom to strenger. Den er laget for å straffe feil prefiks. (Engevik og Engevik, 2022).

I testen settes det en grense for hvor like strengene må være for at de skal regnes som de samme.

5 Resultater

Underveis i prosjektet ble det gjennomført evalueringer. Disse evalueringene ble brukt for å finne mulige forbedringer i løsningen. Det ble også brukt som en måte for å korrigere retningen til prosjektet underveis, slik at oppdragsgiver fikk et resultat som kunne brukes.

5.1 Evalueringsmetode

En viktig del av prosjektgjennomførelsen, er å evaluere hvordan det endelige resultatet stiller opp mot målene satt i starten. Det finnes to hovedsider som evalueres.

5.1.1 Oppdragsgiver

Oppdragsgiveren er brukt som en del av evalueringen. I The Lean Startup metodikken er det en testmetode som heter Concierge MVP. Her er tankegangen at det skal lages et produkt basert på en kunde/bruker. Denne oppgaven hadde en oppdragsgiver som gruppen var concierge for. Tilbakemeldinger fra oppdragsgiver ble brukt til å evaluere hvordan iterasjonen hadde blitt løst, og hva som var neste skritt i utviklingen. Hele veien ble iterasjonene tilpasset oppdragsgiverens sitt behov. (Ries, 2019)

5.1.2 Nøyaktighet

Det som var hoved testmetoden for å se om utviklet løsninger fungerte, var å se om programmet som ble utviklet ga forventede verdier. For å gjøre dette, ble det manuelt registrert i et testprogram hva som var forventet verdi på tittel, komponist, arrangør og stemmene.

Disse verdiene ble brukt for å måle opp mot algoritmen. Basert på dette er det mulig å vurdere om målene ble oppnådd. Resultatene gjør det også mulig å finne feil som løsningen gjør. Dette gjorde det mulig å korrigere for disse feilene.

Evalueringen hadde ekstra fokus på om løsningen fant riktig boks, og ikke nødvendigvis om det som ble returnert var helt riktig. Dette har med begrensninger knyttet til OCR-en og fasen prosjektet ble avsluttet. Prosjektet hadde ikke kommet til finjusteringen av teksten.

5.2 Evalueringsresultat

5.2.1 Oppdragsgiver

Underveis i prosjektet, var det ukentlig møte med oppdragsgiver. Her ble statusen og retningen for prosjektet diskutert. Det ble avklart hvilke behov oppdragsgiveren hadde og hvordan disse skulle tilpasses.

Blant annet var det diskusjon rundt hva som skulle returneres. I starten var det et ønske at API-et skulle returnere en standardisert stemme. Etter hvert ble det klart fra oppdragsgiveren sin side at dette ikke lenger skulle være et mål.

Det ble også lagt inn kommentarer om hvordan JSON-filen som skulle formateres, og et ønske om å kunne oppdatere instrumentlisten som brukes av programmet.

Ved slutten av prosjektet fikk oppdragsgiver mulighet til å se resultatet. Her kom det positive tilbakemeldinger. Slik det ble presentert, var dette et produkt som kunne brukes. Disse tilbakemeldingene tar utgangspunkt i en demonstrasjon. Den tidligere gruppen fikk ifølge rapporten positiv tilbakemelding, selv om den aldri ble tatt i bruk. Det er en mulighet for at oppdragsgiveren etter viere testing, finner at det ikke er en løsning som dekker deres problem.

5.2.2 Nøyaktighet

Som en del av underveisevalueringen, var testingen opp mot forventet verdi viktig. Notesettene ble brukt underveis for å si hvor bra løsningen var, og hvilke punkter som måtte forbedres.

Et problem med dette resultatet, er at programmet har blitt tilpasset notesettene som ble brukt under utviklingen. Her er det mulig å gjøre det som fungerer bra for disse notesettene, ikke nødvendigvis for et annet utvalg. Derfor ble det til slutt testet på et testsett, en ide fra maskinlæring. Et notesett som altså ikke ble sett på under trening. Den samme testen på to forskjellige sett viser at løsningen faller i nøyaktighet. Dette er som forventet.

Informasjon	Trening	Test
Tittel	98%	95%
Komponist	85%	81%
Arrangør	83%	81%
Sett med riktig stemme	81%	57%
Alt riktig	58%	33%
Sider med riktig stemme	98%	94%

Tabell 7:Viser forhold mellom treffsikkerheten på treningsdata mot testdataen

Observasjonen viser at programmet ikke klarer å finne alle stemmene i flere notesett enn tidligere. Fallet fra 81% til 57% er stort. For å utforske disse feilene, ble det gjort en manuell gjennomgang for å finne hvilke feil løsningen gjorde. Blant de 21 testsettene, var det mangler i 14.

Notesett	Mangel
1	*Komponist* - Komponisten ligger over tittelen
2	*Komponist* - Stempel med «original» over komponist
3	*Arrangør* - Arrangør på venstre side
4	*Arrangør* - Ingen tekst foran arrangøren *Stemme* - «Piano» ikke i stemmeliste - Flagget riktig
5	*Komponist* - Tittelen blir komponist. Arket er tosidig
6	*Stemme* - «Soloist» ikke funnet - Flagget feil
7	*Stemme* - «Soloist» ikke funnet - Flagget feil *Tittel* - Bruker tekst under tittel. En boks rundt en stor A er grunnen.
8	*Stemme* - «Soloist» ikke funnet - Flagget feil
9	*Stemme* - «Soloist» ikke funnet - Flagget feil
10	*Stemme* - «Stemmeboksen» er for langt til høyre. Arket er litt rotert.
11	*Stemme* - Blank side blir satt til forrige. Skal settes til «annet»
12	*Arrangør* - Starter med «Transcribed by» *Stemme* - Partituret er horisontalt, stemmen vertikal. Stemmene blir til «annet»
13	*Komponist* - «Text by» over komponist
14	*Arrangør* - Starter med «Transcribed by» *Stemme* - Noen stemmer blir til «annet»

Tabell 8: Viser va som manglet på notesetten som var testet

Under evalueringen, ble det klart at det var mangler med testsettet. Disse ble tydeligere under gjennomgangen. Notesettene som ble testet på var ikke særlig varierte. Notesett 6, 7, 9 og 10 er alle av samme arrangør, og utseendet er like, og inneholder de samme feilene. Notesett 12 og 14 er like, bare på den ene er stemmene rotert.

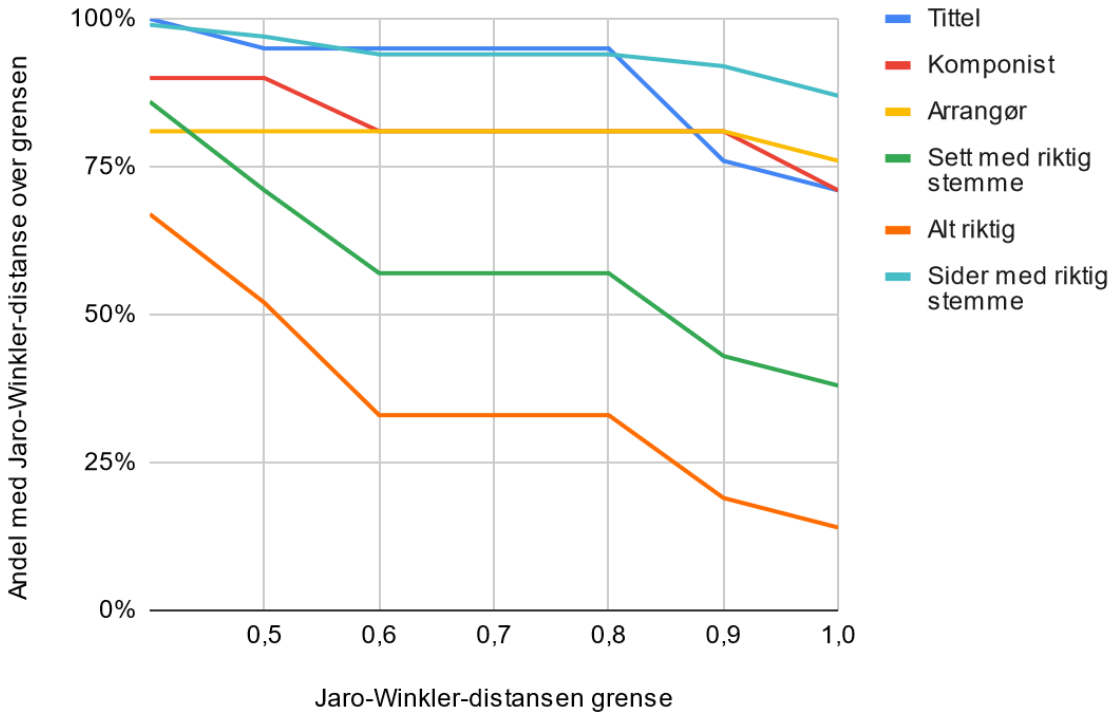
Ulempen med at det ikke er mer variasjon gjør at feil blir amplifisert, samtidig som modellens evner kan bli overdrevne. Det er en forutsetning for evalueringen, at notesettene som oppdragsgiveren kom med var representative for det som ble satt inn i systemet deres. I dette tilfellet var det et problem at 6 av de 21 notesettene som ble testet (29%) var av samme arrangør, og hadde like oppsett. Dermed ble de samme feilene gjentagende. Ved å legge til to instrumenter - «soloist» og «piano» - i instrumentlisten, økte antallet notesett der alle stemmene var riktige fra 57% til 81%.

Informasjon	Justert treff
Tittel	95%
Komponist	81%
Arrangør	81%
Sett med riktig stemme	81%
Alt riktig	48%
Sider med riktig stemme	96%

Tabell 9: Oppdatert treffsikkerhet

Under gjennomgangen fant ikke løsningen «Piano» flagget riktig som en forventet førsteside. Det gjorde ikke «Soloist». Dette har med at maskinlæringsmodellen ikke klarte å predikere stemmen som en førsteside. Løsningen burde ha flagget dette som en førsteside ettersom sidetypen er den første «note»-siden etter en partiturside.

Under evalueringen, er det tatt noen aktive valg. Dette har med at i løpet av prosjektet, var hovedmålet å finne riktig «boks» og ikke helt riktig tekst. Hvor grensen settes endrer evalueringen.



Figur 22: Viser forhold mellom treffsikkerhet ved jaro-winkler-distansen grense

Her har det blitt valgt å bruke 0,8 som grense. Denne grensen ble brukt, ettersom det er høyt nok til at hvis en finner feil boks, blir dette satt til feil. Dette er noe som er observert i manuelle gjennomganger. Den vil heller ikke straffe mindre feil. I dette tilfellet går tittelen fra 76% til 95% når grensen settes ned fra 0,9 til 0,8. Dette er fordi det er en del notesett i testsettet der tittelen starter med tall, f.eks. «07 - My Favorite Things». Det er dette løsningen vil returnere, selv om den ønskede verdien er satt til «My Favorite Things».

En utfordring som dukket opp underveis i utviklingen, var at det ønskede resultatet ikke alltid var like lett å finne. Det kunne være utfordringer med at enkelte stemmer hadde g- og f-nøkler som OCR-en ikke klarte å fange opp, vises i Figur 23 og Figur 24 som de siste symbolene bak «trombone». I et optimalt tilfelle, ville disse ha vært med. Dette er noe oppdragsgiver har nevnt at ikke er viktig.

Figur 23: eksempel på stemme med g-nøkkel. Hentet i fra Lord of the rings (Shore og Valta, ingen dato)

Figur 24: eksempel på stemme med f-nøkkel. Hentet i fra Lord of the Rings (Shore og Valta, ingen dato)

Det er tilfeller der OCR-en gjør feil. Det er flere tilfeller der «B♭» ble tolket som bare «B» istedenfor «Bb». I et tilfelle, var instrumentet og tallet i ett ord. Da ble «Trombone1» (med et ett tall) til «Trombonel» (med en L). Slike feil påvirker ikke scoren i dette tilfellet.

5.3 Prosjektresultat

Det kan være vanskelig å konkludere med hvor bra resultatene er. Det er flere faktorer involvert enn antatt i en tidlig fase av prosjektet. Ut ifra forenklingene som er gjort, ser det ut til å oppnå enkelte av kravene.

Evalueringsresultatene bidrar til å vise at problemstillingene ble oppnådd. Ved bruk av concierge MVP testing metoden, klarte gruppen sammen med oppdragsgiver å levere et API som kunne forenkle prosessen med å registrere stemmen, tittel, komponist og arrangør fra noteark.
Målsetting ble ikke nådd ut fra evalueringsresultatene. Målsetningen var å få riktig data på hele notesettet i 80-90% av notesettene. Det ble ikke oppnådd ut fra målingene. Grunnen til dette var at når målsettingene ble satt, var det en antagelse at notesettene med feil, ville inneholde mesteparten av feilene, og ikke at flere av notesett ville ha en og en feil som vises i Tabell 8. Det var dermed en feil antagelse når målet ble satt.

Det er stor usikkerhet til målingene ettersom test-settet er lite.

5.4 Prosjektgjennomføring.

Prosjektgjennomføringen gikk delvis etter planen. I første iterasjon ble det arbeidet godt og målene ble oppnådd. I iterasjon 2 satt gruppen for høye mål. Testmetoden var etter planen noe som ikke skulle prioriteres. Dette endte opp med å ta mesteparten av tiden som førte til at målene for iterasjonen ikke ble oppnådd. Derfor var utviklings målet fra iterasjon 2 overført til iterasjon 3. Ettersom prosjektet lå bak planen, ble det valgt å legge til flere mål i iterasjon 3. I iterasjon 3 var det bare utviklingsmålene fra iterasjon 2 som ble oppnådd. Sykdom og mangel på tid førte til at det var vanskelig å oppnå alt. Iterasjon 4 fikk målene fra iterasjon 3. Algoritmeløsningen ble utviklet. Denne var ikke god nok i forhold til prosjektmålene, men et godt utgangspunkt for videre arbeid. Ettersom maskinlæringsløsningen ikke hadde kommet skikkelig på plass, ble det tatt en avgjørelse om å satse på algoritmeløsningen videre.

Når planlegging for iterasjonene ble satt opp, ble det kun satt opp 4 iterasjoner. I denne planen var det rom for å legge til ekstra iterasjoner, om nødvendig. Det ble lagt til 2 nye iterasjoner på 1 uke. I iterasjon 5 var målet å gjøre algoritmeløsningen bedre og lage en mulighet for å oppdatere instrumentlisten. Dette ble oppnådd, samtidige ble det startet et nytt maskinlæringsprosjekt. Iterasjon 6 gikk ut på å lage et «flagging»-system, fullføre maskinlæremodellen og finpusse koden. Dette ble oppnådd og ble det levert et fullstendige produkt.

	Mål	Oppnådd
Iterasjon 1	Få opp et API som kan ta in PDF-filer. Få den gjennom OCR-en som henter ut tekst fra PDF-filene.	Alle mål ble oppnådd. Google sine løsninger Cloud Storage for lagring av data og OCR-verktøyet Vision API ble brukt.
Iterasjon 2	Lage en løsning som kan hente ut informasjon fra en noteside som ikke hadde framside eller partiturside som første side. Det trenger ikke å være riktig data som kom ut. Lage en testmetode.	Målet å lage testmetode ble oppnådd. Mye tid som var brukt for å gjøre det enklere å teste for algoritme og maskinlærens løsning. På grunn av mye tid brukt på testing, ble ikke utviklingsmålet fullført.
Iterasjon 3	Lage 2 enkle, men fungerende løsninger av programmet. Den ene algoritmen og den andre maskinlæring. Gitt tid for å videre arbeide med testmetoden.	Det ble ikke laget 2 løsninger. Sykdom og andre uforutsette hendelser førte til at det stoppet utvikling. Ved slutten av iterasjonen hadde det blitt laget en enkel algoritmeløsning som kunne vise enkelt hvordan det fungerte, tankegangen vår og hvordan JSON-filen skulle se ut.
Iterasjon 4	Målene fra iterasjonen 3 ble flyttet over til denne.	Fikk til en generell algoritme løsning som var noe som kunne bygge videre på. Arbeid med NER-løsning ble avsluttet. Algoritmeløsningen viste seg å ha mer potensial.
Iterasjon 5	Videreutvikle algoritme løsningen. Lage et API-endepunkt som kan oppdatere stemme databasen. Oppstart av nytt maskinlæringsprosjekt, der det skulle klassifisere sider.	API-endepunktet ble laget. Algoritme løsningen ble bedre. Maskinlæring prosjekt kom i gang
Iterasjon 6	Fullføre maskinlæringsmodellen. Lage flagging-system. Fin pusse kode og dokumentasjon.	Maskinlæringsmodellen kom på plass. Kunne brukes til å registrere hvilken type side det var som ble lest av. Hjalp algoritme løsningen veldig mye. Flagging-system på mulige feil sider og andre feil. Koden ble gjort klar for levering ved å dokumentere og finpusse den.

Tabell 10: tabell for iterasjon mål og hva som ble oppnådd

6 Diskusjon

Produktet svarer til flere av ønskene til oppdragsgiveren. Tilbakemeldingene indikerer at løsningen er tilpasset deres behov. Målet med prosjektet var å utvikle et API som kunne returnere informasjon om innsendte notesett i form av en JSON-fil. Selv om løsningen ikke oppnår den forventede suksessraten på 80-90% for nøyaktig gjenkjenning av notesett, er resultatet likevel lovende. Det finnes en mulighet for å benytte seg av løsningen i prosessen ved å splitte notesettene opp i ulike stemmer.

Et av oppdragsgiveren sine ønsker, var å ha en form for flagging i løsningen. Dette ønsket er implementert og gir mulighet for å identifisere feil som løsningen gjør. Det er imidlertid viktig å merke seg at denne funksjonen ikke er feilfri. Under testingen ble det oppdaget at den brukte feil flagg i en situasjon som tidligere ikke hadde blitt tatt i betraktning.

6.1 Utvikling

6.1.1 Dokumentasjon

Et aspekt som det har blitt fokusert på underveis, er kodekvalitet. Det var viktig at koden ble dokumentert med kommentarer. Det ble også brukt «typing». Dette kan hjelpe med å gjør det tydeligere hva de forskjellige delene av koden gjør. Det er også lagt til enkel dokumentasjon utover kommentarene som finnes i koden. Det har vært et mål at koden skal være av en slik kvalitet at det kan være mulig å benytte hele eller deler av koden i eventuelt videre arbeid.

6.1.2 Tester

En mulig svakhet med programmet, er mangelen på unit-tester. Selv om programmet ikke har unit-tester, er det laget tester som kan brukes. Blant annet er det en test som kjører hele algoritme delen for å se at den fungerer. Det finnes også tester som kjører på alle notesettene som er lagt inn i testsystemet. Hvis det er feil i flyten, blir den oppdaget her. En av feilene som ble oppdaget under kjøringen av dette, er knyttet til posisjonene til boksen. Det er tilfeller der det ikke blir oppgitt en «x» eller «y» i koordinaten. Det var en tidlig antagelse at disse alltid er med. Ved hjelp av unit-tester hadde det vært mulig å teste mindre deler av programmet. Et av problemene er at det ikke er en standard input, og er derfor vanskelig å forutse hvilke tilfeller som dukker opp.

6.1.3 Labelingsystem

Det ble laget et program som kan brukes til å manuelt markere hva som er ønsket at produktet skal returnere. Det har vært en nyttig del for å kunne evaluere resultatene til løsningen fortløpende. Labelingprogrammet er av lavere kvalitet enn løsningen. Det har flere mangler og koden er ikke strukturert på en slik måte at den er lett å jobbe med. Denne

delen av prosjektet er ikke direkte en del av løsningen og flere av disse aspektene har ikke blitt prioritert.

6.2 Resurser

En av svakhetene med løsningen er at den er basert på en begrenset mengde data. Gruppen fikk til sammen ca. 70 notesett fra oppdragsgiver som løsningen har basert seg på. I utgangspunktet skulle de notesettene være tilfeldig valgt slik at variasjonen ble stor og representativ for det løsningen skal brukes til. En utfordring her er at det er flere eksempler på at variasjonen er manglede.

6.2.1 Konsekvenser

Den manglende variasjonen har som konsekvens at maskinlæringsmodellen vil gjøre feil som kunne vært unngått med en større datamengde. Hvis den predikerer «annet» på sider som ikke er annet, vil det ha store konsekvenser for hva modellen predikerer. Her er det problemer om oppsettet er noe unormalt. Dette ble sett i testsettet. Mengden data som maskinlæringsmodellen er trent på, er ikke veldig variert, noe som gjør at den kan ende opp med å predikere feil ting.

Datamengden har også skapt problemer med andre aspekter i løsningen. Løsningen er ikke laget for å kunne tolke PDF-er der det er to sider på et ark. Den er heller ikke laget for å kunne tolke noter der siden er liggende. De eneste eksemplene av disse finnes i testsettet. Det finnes trolig andre uregelmessigheter som hadde dukket opp med en større datamengde.

En konsekvens av den begrensede mengden notesett er at det er stor usikkerhet knyttet til målingene. Dette er en begrensing som kan gjøre det vanskelig å si noe om hvor bra modellen vil gjøre det i virkeligheten, og dermed si noe om resultatet.

6.3 Nøyaktighet

Løsningen er spesielt svak på arrangører og komponister. Dette har delvis med at disse, sammen med tittel, ikke ble høyt prioritert. Noe av grunnen til dette er at oppdragsgiveren nevnte at det først og fremst var stemmen som var det viktigste. Dette er delvis fordi det tar mindre tid å hente ut komponist, arrangør og tittel, enn det er å splitte sidene opp i ulike stemmer.

Stemmen har også begrensninger. Disse er til en viss grad knyttet til at det er en stemmeliste som benyttes. Som tidligere nevnt, viser tester gjort at det ikke er nødvendig å ha alle instrumentene i listen for at den skal oppnå gode resultater. Det er uansett viktig at denne listen er fullstendig ettersom det ikke er alle situasjoner der løsningen klarer seg uten dette. Det er antatt at en mer fullstendig liste, kan øke treffsikkerheten.

Det er stemmene som løsningen treffer best på. Her er det en fordel at det finnes en liste over instrumenter som gjør det mulig å lokalisere boksen. Denne fordelen finnes ikke for tittel, arrangør og komponist. Dette reflekteres i at treffprosenten på disse (95%, 81% og 81%) er lavere enn på stemmene hvis en tar utgangspunkt i enkeltsider (96%).

Selv om det finnes aspekter med programmet som ikke er optimale, er det fremdeles en fremgang fra det tidligere prosjektet. Det tidligere prosjektet hadde flere begrensinger knyttet til seg. Basert på testene beskrevet i rapporten, virker det som at løsningen fungerte dårligere. Noe av forbedringen på tittel, komponist og arrangør, er trolig knyttet til bruken av sidetyper. Å bruke sidetyper til å bestemme hvilke sider som skal ses på, virker til å fungere bra. Et annet aspekt er knyttet til funn av stemmen. Her var målet noe annerledes i den forrige oppgaven. Disse kan derfor ikke sammenlignes direkte.

6.4 Prosjektgjennomføring

En viktig del av utviklingen var bruken av The Lean Startup. Metoden var viktig i utviklingen ettersom den bidro til at prosessen ble agil og iterativ. Dette gjorde det mulig å tilpasse prosessen underveis. Dette gjelder både tilpasninger basert på oppdragsgiverens tilbakemeldinger og fremgangen i prosjektet. Muligheten til å endre planen underveis, viste seg å være nødvendig i tilfeller der ting som sykdom gjorde at arbeidet ble forsinket.

Det var flere valg knyttet til prosjektgjennomføringen som har begrenset resultatet. Noe av dette er knyttet til ønske om å bruke en NER-modell for å finne den relevante informasjonen. Ved en ny gjennomgang av prosjektet ville dette blitt prioritert tidligere. Risikoen knyttet til vansker med en slik bruk av maskinlæring var kjent.

For å forminske konsekvensene av denne risikoen, ble det valgt å ta to tilnærminger til oppgaven. En maskinlæringsbasert og en algoritmebasert metode. Dette førte til at konsekvensene ble mindre enn de kunne ha vært. Dette skjedde i iterasjon 3 og 4, men det er tydelig at iterasjon 3 og 4 hadde begrensede resultater. Dette er en periode på 5 uker. En mer effektiv tidsbruk som å prioritere den ene fremgangsmåten over den andre, kunne føre til bedre og raskere resultater på ett av områdene.

6.4.1 Oppdragsgiver

Det at det har vært regelmessig kontakt med oppdragsgiver som har hjulpet med at produktet svarer på de utfordringene oppdragsgiver står overfor. Blant annet ble målet med å returnere en standardisert stemme fjernet etter at dette hadde vært tema på flere av møtene. Dette hjalp oppdragsgiver med å reflektere rundt hva det egentlige behovet var. Det ble klart at det var bedre at programmet returnerte det som stod, fremfor å gjette på betydningen av det som stod.

Det ble også klargjort andre aspekter med prosjektet som at hastigheten av programmet ikke var et problem. Dette var noe som ble trukket frem tidlig i prosjektet, ettersom at gruppen så på dette som en svakhet med den tidligere løsningen. Oppdragsgiveren hadde

derimot ikke den samme oppfattingen. Han mente det var viktigere å få en nøyaktig løsning fremfor en rask en.

6.5 Bruk

Det er noe usikkerhet rundt hvordan oppdragsgiver ønsker å benytte seg av løsningen. I utgangspunktet skulle stemmene returneres på et standardformat. Dette er nå ikke lenger situasjonen. Hvordan oppdragsgiveren har tenkt til å splitte basert på det som returneres er usikkert. Dagens løsning går ut på at instrumenter/stemmer velges fra en liste når notesettene deles opp. Dette skal ikke gjøres her. En mulighet er at den som skal dele ut notesettet, manuelt sier hvilken gruppe som hører til det som ble satt til stemme.

Som diskutert tidligere, er det stor usikkerhet knyttet til hvor bra løsningen er. Det er derfor viktig at den ikke blir satt i produksjon uten videre testing på hvilke begrensinger som muligens eksisterer. Om dette ikke gjøres, kan det ha konsekvenser for produktet.

7 Konklusjon og videre arbeid

Prosjektets hovedmål var å lage et API for å kunne registrere hvilke stemmer som er knyttet til en side som kunder og Styreportalen selv kunne bruke for å forenkle digitalisering av notesett. Andre mål som var satt var å få ut annen relevant data som tittel, komponist og arrangør. Målet var å oppnå 80-90% treffsikkerhet på notesettene.

I prosjektet ble det laget et API som oppfyller mange av kravene til oppdragsgiver. Treffsikkerheten for hver enkel data ser ut til å kunne være over 80%. Det er usikkerhet rundt dette tallet. Det den ikke fikk til, var å ha over 80% av notesettene der all dataen var riktig.

7.1 Måloppnåelse

Hovedmålet med å få 80-90% av all data på hvert sett riktig, ble ikke oppnådd. Når det ble satt som mål tidlig i prosjektet, var det ideen at det fantes notesett som det ikke gikk an å lese. Derfor synes det som en rimelig grense at 1 av 10 var uleselig. Det viste seg etter at API-et ble laget, at dette ikke var tilfellet. Istedenfor at hele notesett var feil, var det små feil på flere sett. Det å måle hvor mange som ble riktig, var også vanskeligere enn først antatt.

Et delmål var å lage et system som ville flagge dersom API tror at det har gjort feil. Dette skal hjelpe med å identifisere mulige feil ved hjelp av å fortelle hva løsningen har gjort. Dette delmålet ses på som oppnådd.

Styreportalen var fornøyd med API-et som ble laget. Oppdragsgiveren sa at dette var noe de kommer til å bruke i utviklingen med å få på plass en løsning som forenkler prosessen med å legge inn nye notesett i *Notearkiv*. Det som trengs, er å øke treffsikkerheten for hele notesett.

7.2 Mulig videre arbeid

Det finnes flere ting som kan forbedre løsningen som er laget. Et av aspektene med prosjektet som aldri ble skikkelig fremgang på, var bruken av maskinlæring til å finne den delen av teksten fra OCR-en som hører til tittel, stemme, arrangør og komponist. Det er vanskelig å forutsi hvor nyttig en slik modell kan være. En maskinlæringsmodell vil trolig kunne håndtere flere usikre tilfeller enn det en algoritmebasert metode vil, men det kan også oppstå at den er dårligere på mer standard oppsett.

En annen mulighet ville vært og hatt en maskinlæringsmodell som kunne ha blitt brukt for etterprosessering av tekstboksen som ble funnet. I flere tilfeller inneholder tekstboksene ekstra tekst. Til tider returnerer OCR-en feil i teksten som gjør det vanskelig å få korrekt resultat. Her kunne det vært mulig å få gjort mer etterprosessering.

Et aspekt med API-et som er laget er hvor lang tid det bruker. Dette har vært en diskusjon gjennom prosjektet med oppdragsgiver. Tilbakemeldingen har vært at dette aspektet ikke skal prioriteres. Selv om dette aldri ble prioritert, ble det gjort refleksjoner rundt hvordan dette kunne ha blitt forbedret. I utgangspunktet er det OCR-en som tar tid. Dette inkluderer opplasting til Cloud Storage og Vision API.

En mulig måte å øke hastigheten på ville vært å finne alternative tilnærminger for å hente ut teksten. For PDF-filer er det nødvendig å laste opp filene, men det er mulig å sende enkeltbilder til Vision API uten å laste dem opp. Det er en mulighet for at dette kunne ha økt hastigheten.

Et annet alternativ ville vært å undersøke om det er et skannet PDF-dokument eller ikke. For skannede PDF-dokumenter, lages sidene som bilder. Hvis PDF-filen er digital, finnes det verktøy som kan hente ut teksten i dokumentet. Denne teksten kan inkludere metadata som størrelse, font og posisjon. Dette kunne trolig ha fjernet noen av feilene som OCR-en har gjort. Tester i en tidlig fase av prosjektet, viste at dette i flere tilfeller ville vært raskere. Dette ble ikke prioritert, ettersom en OCR-basert metode for uthenting av tekst er mer generell.

8 Figur- og tabell-liste

9 Referanser

Bækkelund, N.G. (2022) ‘Innovasjonsmetodikkar Design thinking, lean startup og prototyping’. ING303 Systemtenking og innovasjon for ingeniørar.

BandMusic PDF Library (ingen dato) ‘BandMusic PDF Library – Preserving & Sharing Music from the Golden Age of the Town Band’. Tilgjengelig: https://bandmusicpdf.org/ (Hentet: 8 mars 2023).

Bernie, B. *et al.* (ingen dato) ‘Sweet Georgia Brown’. Editions Marc Reift.

Boccara, J. (2021) ‘GRASP: 9 Must-Know Design Principles for Code’, *Fluent C++*, 23 June. Tilgjengelig: https://www.fluentcpp.com/2021/06/23/grasp-9-must-know-design-principles-for-code/ (Hentet: 7 mars 2023).

BPMN 2.0 Symbols - A complete guide with examples. (ingen dato) *Camunda*. Tilgjengelig: https://camunda.com/bpmn/reference/ (Hentet: 19 mai 2023).

Cheng, T. (2022) *Intro to Google Vision OCR - OCR, APIs, and Examples, Nanonets AI & Machine Learning Blog*. Tilgjengelig: https://nanonets.com/blog/google-cloud-vision/ (Hentet: 8 mars 2023).

Document AI (ingen dato) *Google Cloud*. Tilgjengelig: https://cloud.google.com/document-ai (Hentet: 20 mai 2023).

Engevik, A. og Engevik, J.M. (2022) *Utvikling av kategoriseringsverktøy for PDF-notesett*. Bachelor thesis. Høgskulen på Vestlandet. Tilgjengelig: https://hvlopen.brage.unit.no/hvlopen-xmlui/handle/11250/3020821 (Hentet: 13 januar 2023).

Evart, T. og Jerker Johansson (ingen dato) ‘Kosterflickorns visa’. Norsk Noteservice As.

Explosion (ingen dato) *Linguistic Features · spaCy Usage Documentation, Linguistic Features*. Tilgjengelig: https://spacy.io/usage/linguistic-features (Hentet: 21 mai 2023).

fastai (ingen dato a) *fastai - Learner, Metrics, Callbacks*. Tilgjengelig: https://docs.fast.ai/learner.html#learner.export (Hentet: 20 mai 2023).

fastai (ingen dato b) *fastai - Vision widgets*. Tilgjengelig: https://docs.fast.ai/vision.widgets.html (Hentet: 21 mai 2023).

GeeksforGeeks (2021) ‘What are the key features of Node.js ?’, *GeeksforGeeks*, 28 October. Tilgjengelig: https://www.geeksforgeeks.org/what-are-the-key-features-of-node-js/ (Hentet: 6 mars 2023).

Géron, A. (2019) *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*. Second edition. Beijing [China] ; Sebastopol, CA: O’Reilly Media, Inc.

Google (ingen dato a) *Cloud Functions, Google Cloud*. Tilgjengelig: https://cloud.google.com/functions (Hentet: 6 mars 2023).

Google (ingen dato b) *Cloud Run: Container to production in seconds, Google Cloud*. Tilgjengelig: https://cloud.google.com/run (Hentet: 18 mai 2023).

Google (ingen dato c) *Detect text in files (PDF/TIFF) | Cloud Vision API, Google Cloud*. Tilgjengelig: https://cloud.google.com/vision/docs/pdf (Hentet: 10 mai 2023).

Google (ingen dato d) *Quickstart: Deploy a Python service to Cloud Run | Cloud Run Documentation, Google Cloud*. Tilgjengelig: https://cloud.google.com/run/docs/quickstarts/build-and-deploy/deploy-python-service (Hentet: 18 mai 2023).

Gross, W., Lawrence, J. og Morrissey, N.J. (ingen dato) 'Tenderly'. Edwin H. Morris and Company, Inc.

Halvorsen, J. og Laurendeau, L.P. (ingen dato) 'Einzugsmarsch der Bojaren'. New York: Carl Fischer.

Howard, J., Gugger, S. og Chintala, S. (2020) *Deep learning for coders with fastai and PyTorch: AI applications without a PhD*. First edition. Sebastopol, California: O'Reilly Media, Inc.

Hummel, G. (ingen dato) *Summary - Overview of Google Cloud Platform Course*. Tilgjengelig: https://cloudacademy.com/course/overview-google-cloud-platform-1245/summary/ (Hentet: 6 mars 2023).

JavaTpoint (ingen dato) *Advantages and disadvantages of Java - Javatpoint, www.javatpoint.com*. Tilgjengelig: https://www.javatpoint.com/advantages-and-disadvantages-of-java (Hentet: 6 mars 2023).

Meyerbeer, G., Ptikin, B. og Trevitz, S. (ingen dato) 'Fackeltanz in Bb'. BandMusic PDF Library.

Nagelhus, L.A. (2008) *Den store musikkordboka*. Oslo: Norsk musikkforl.

Pallets (ingen dato) *Welcome to Flask — Flask Documentation (2.3.x)*. Tilgjengelig: https://flask.palletsprojects.com/en/2.3.x/# (Hentet: 19 mai 2023).

Prasanna (2022) 'Advantages and Disadvantages of Python | Python Language Advantages, Disadvantages and Its Applications', *A Plus Topper*, 7 januar. Tilgjengelig: https://www.aplustopper.com/advantages-and-disadvantages-of-python/ (Hentet: 6 mars 2023).

pypdfium2-team (ingen dato) 'pypdfium2: Python bindings to PDFium'.

Python Software Foundation (ingen dato) *threading — Thread-based parallelism, Python documentation*. Tilgjengelig: https://docs.python.org/3/library/threading.html (Hentet: 10 mai 2023).

Ries, E. (2019) *The Lean Startup* (1 vol). 1st edn. Great Britain: Penguin Buisness.

Shore, H. og Valta, J. (ingen dato) 'The Lord Of The Rings The Fellowship'. Editions Marc Reift.

Sterri, M.T., Taraldset, L.M. og Thorsteinsson, H.B. (2023a) 'Tolking av PDF-notesett og gjenkjenning av stemmer. - Kravdokumentasjon'.

Sterri, M.T., Taraldset, L.M. og Thorsteinsson, H.B. (2023b) 'Tolking av PDF-notesett og gjenkjenning av stemmer. - Prosjekthåndbok'.

Sterri, M.T., Taraldset, L.M. og Thorsteinsson, H.B. (2023c) 'Tolking av PDF-notesett og gjenkjenning av stemmer. - Systemdokumentasjon'.

Sterri, M.T., Taraldset, L.M. og Thorsteinsson, H.B. (2023d) 'Tolking av PDF-notesett og gjenkjenning av stemmer. - Visjonsdokument'.

Styreportalen (ingen dato) *Styreportalen AS: Om | LinkedIn*. Tilgjengelig: https://www.linkedin.com/company/styreportalen/about/ (Hentet: 21 mai 2023).

Zimmer, H., Oftedal, H. og Furholt, K. (ingen dato) 'Intersteller Melody'.

10 Vedlegg

10.1 Prosjekthåndbok

Prosjekt håndboken inneholder Gant-diagram, risiko tabell, risiko matrise, møteinnkallinger og referat, og timelister med statusrapport (Sterri, Taraldset og Thorsteinsson, 2023b).

10.2 Systemdokumentasjon

Dokument med flere figurer med oversikt over systemet. En beskrivelse av API-et også med. Ulike sider som installasjon, dokumentasjon og testing blir beskrevet. Her kan er det mulig å få tilgang til koden under sensur. Lenken til demosiden finnes også her (Sterri, Taraldset og Thorsteinsson, 2023c).

10.3 Kravdokumentasjon

Beskriver flere av kravene som oppdragsgiver presenterte i en tidlig fase av prosjektet (Sterri, Taraldset og Thorsteinsson, 2023a).

10.4 Visjonsdokument

Dokument med ulik informasjon slik det ble presentert i en tidlig fase av prosjektet (Sterri, Taraldset og Thorsteinsson, 2023d).