Høgskulen på Vestlandet

BACHELOROPPGAVE

Utvikling av kategoriseringsverktøy for PDF-notesett

Development of categorising tool for sets of notes in PDF-format

Anders Engevik

Johan Magnus Engevik

Dataingeniør
Fakultet for ingeniør og naturvitenskap, institutt for datateknologi, programutvikling og drift av datasystem
Veileder: Atle Birger Geitung
23.05.2022

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Utvikling av kategoriseringsverktøy for PDF-notesett	*Dato:23.05.22*
Forfatter(e): Anders Engevik, Johan Magnus Engevik	*Antall sider u/vedlegg:* *34* *Antall sider vedlegg: 1*
Studieretning: *Dataingeniør*	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Atle Birger Geitung	*Gradering:* ingen
Merknader:	

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

Sammendrag:

Dette prosjektet handler om å utvikle en programvare for kategorisering av notesett som Styreportalen.no kan ta i bruk i sine systemer. Dette ble gjennomført ved å lage et program i Firebase som bygde på Google Vision og Google Cloud Functions. Programmet viste seg å ha en høy prosentandel korrekt kategoriserte notesett, men det er fortsatt muligheter for optimering.

Stikkord:

Google Vision	Google Cloud Functions	Firebase

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

Denne rapporten omhandler bachelorprosjektet «Utvikling av kategoriseringsverktøy for PDF-notesett. Oppdraget for prosjektet ble gitt av Styreportalen AS, og ble utført av Anders Engevik og Johan Magnus Engevik.

Vi ønsket å gå inn i dette prosjektet fordi vi begge har vært aktive i korpsmiljøet gjennom flere år, og på den måten har interesse for og erfaring med bruk av notesett. Nye systemer og utforming av utviklingsspråk innenfor dette området har vært både lærerikt og utfordrende.

Vi vil takke Styreportalen AS for muligheten til å jobbe på et spennende prosjekt som dette. I tillegg vil vi rette en spesiell takk til Stian Sømoen for gode innspill, kommentarer og forslag til hvordan vi kunne løse utfordringer underveis i prosessen.

Til slutt vil vi rette en spesiell takk til vår veileder gjennom prosjektet, Atle Birger Geitung, for hans tilbakemeldinger og hjelp til å strukturere rapporten.

Anders Engevik og Johan Magnus Engevik

INNHOLDSFORTEGNELSE

1 INNLEDNING

1.1 Kontekst

Styreportalen er en gründerbedrift som brukes av mange korps i Norge pr. I dag (Sømoen, 2022). Bedriften startet opp i 2016 og holder til på Nesttun utenfor Bergen. Den dag i dag har de 6 ansatte (Sømoen, 2022) som jobber med å utvikle programvare for frivillige organisasjoner. Hovedproduktene deres er Styreportalen.no, Billett.no og Program.no. Styreportalen er en nettside for korps, band o.l. der det åpnes for nettbasert notearkiv og kalender for bedre administrasjon. Billett.no er en nettside for billettsalg til konserter o.l., mens program.no er en nettside som viser oversikt over kommende arrangement.

Styreportalen.no er en nettside som brukes av korps rundt om i Norge for arkivering og håndtering av deres notearkiv. Her kan de laste opp pdf-er av sine noter, kategorisere dem og sortere dem deretter. Nettsiden har også moduler i sitt dashbord som prosjekter, kalender og personer for økte administrasjonsmuligheter.

Styreportalen går i dag igjennom en stor ekspandering av sin bedrift, mer spesifikt styreportalen.no (Sømoen, 2022). Antallet kunder de skal betjene har økt drastisk, noe som medfører en naturlig økning i forventninger til funksjoner i deres programvarer. Derfor har de kommet med et ønske om hjelp fra HVL der tanken er at 4 grupper bachelorstudenter kan hjelpe dem.

1.2 Motivasjon

Gjennom en ny avtale mellom Styreportalen og Norges Musikkorps Forbund (NMF) som ble inngått i 2021, ble det bestemt at alle korps i Norge skal bruke deler av styreportalen.no. Det er mye på grunn av dette at den økte funksjonaliteten er nødvendig. Det kan ta lang tid å kategorisere et notesett. Sømoen hevder, basert på egen erfaring, at kategoriseringen av ett enkelt notesett kan ta 2-3 minutter med effektivt arbeid (Sømoen, 2022). Det er over 1600 korps i Norge i dag som er medlem av NMF (Norges Musikkkorps Forbund, 2022). Når da enhver notearkivar må gjennom denne prosessen for alle notene i et korps arkiv, samler det seg fort mye tid som går med kategorisering da hvert korps kan ha flere hundre notesett, om ikke mer. Derfor er det essensielt å få på plass en automatikk ved kategorisering av noter og prosessen rundt dette.

1.3 Prosjekteier

Prosjekteier er Styreportalen. Prosjektgruppen har ingen eierskap til produktløsningen og den utviklede programvaren. Programvaren bruker også en del andre ferdigutviklede ressurser som vi ikke kan kreve som våre. Den ferdige løsningen vår skal dessuten bli en integrert del av systemet til styreportalen, som vi ikke noen rettigheter til.

1.4 Problembeskrivelse og mål

Som det fremgår av pkt. 1.2, er vår oppgave å lage en funksjon for Styreportalen som skal gjøre det enklere for korps å legge inn nye noter i sitt arkiv. Utviklingen omfatter altså både en sortering basert på skanning, samt en funksjon for arkivering av notesett og ekstrahert informasjon. Samtidig er det et ønske fra Styreportalen å få med tittel og forfatter i denne kategoriseringen. Vi ønsker altså her å få et visst element av automasjon inn i arkiveringsprosessen av notesett. Ifølge Salvendy og Karwowski (Salvendy & Karwowski, 1997) handler automasjon om å øke presisjon og forbedre økonomien til prosesser, samt å redusere brukerens arbeidsmengde. Det er altså viktig at løsningen vår på oppgaven faktisk gjør det enklere og tidsbesparende for brukerne.

Målet vårt er å lage et kategoriseringsverktøy for notesett som Styreportalen kan ta i bruk som en del av deres nettbaserte arkivløsning. Gjennom denne kategoriseringsfunksjonen skal en bruker kunne laste opp ett, eller potensielt flere notesett, som deretter blir automatisk kategorisert ved hjelp av programvaren vi har utviklet. Et automatisk kategoriseringsverktøy vil gjøre det mulig å redusere tiden det tar for en notearkivar å arkivere et korpsnoter betydelig.

Vår problemstilling blir da som følger:

Utvikling av kategoriseringsverktøy for PDF-notesett.

1.5 Oppbygging av rapporten

Denne rapporten er bygd opp av 9 Ulike kapitler. I kapittel 1 tar vi for oss problemstilling, mål, kontekst og motivasjon for oppgaven. Kapittel 2 omhandler beskrivelsen av selve prosjektet, avgrensinger og ressurser som brukes, samt litteratur. I kapittel 3 diskuteres designet av oppgaven. I kapittel 4 legges den detaljerte løsningen frem. Her går vi mer i dybden av valgene vi tar, hvorfor vi gjør dem, samt sammenligninger med andre alternativer. Kapittel 5 beskriver resultatene våre. Her går vi stegvis gjennom evalueringen av resultatene samt en vurdering av dem. I kapittel 6 diskuteres resultatene opp mot løsningen vår. Her ser vi på hva som var bra, hva som kunne vært bedre og hvorfor det var slik. Til slutt, i kapittel 7, kommer konklusjonen. Her dras konklusjonen samt veien videre vil bli formidlet. Videre etter dette kommer kapittel 8 og 9 som gir en oversikt over referanser og vedlegg.

2 PROSJEKBESKRIVELSE

2.1 Praktisk bakgrunn

I dette kapittelet tar vi for oss den praktiske bakgrunnen for prosjektet. Her vil de tekniske kravene til prosjektet som kommer fra oppdragsgiver legges frem og drøftes.

2.1.1 Tidligere arbeid

Styreportalen har i dag en programvare som gir brukere en nettbasert løsning for notearkivering. Her kan brukere laste opp notesett de har, så lenge det er i form av en PDF-fil. I det den blir lastet opp vil programvaren hjelpe bruker å splitte opp notesettet slik at PDF-en blir delt opp etter stemmer i notesettet. Men dette gjøres per i dag ved assistanse av bruker. Styreportalen ønsker at kategoriseringen basert på stemme også skal være automatisk, da det er en tidkrevende prosess å gjøre denne jobben manuelt for en bruker. Nettopp derfor har de spurt oss om å lage en slik løsning som kan bli en del av programvaren i systemet de bruker i dag.

2.1.2 Initielle krav

Styreportalen har lenge tenkt at automatisk kategorisering skal blir en integrert del av deres system. Med en god løsning på dette vil insentivet for å ta i bruk plattformen bli enda større. Bedriften ønsker å ta i bruk Googles Vision API som en del av deres løsning for kategoriseringen (Visjonsdokument, Vedlegg) (Kravdokumentasjon, Vedlegg). Det er et API for uthenting av informasjon fra bilder (Google Developers, 2022). Vision kan blant annet brukes til gjenkjenning av kjente landemerker, gjenkjenning av ansikt til kjente personer, og merke bilder med forhåndsbestemte etiketter. For oss er det Vision sitt API for å gjenkjenne tekst i dokumenter i PDF/TIFF format som er relevant.

Det har ikke blitt utviklet noen løsning for denne problemstillingen tidligere av bedriften. Men siden bedriften allerede bruker React og Node.js, var det ønskelig at arbeidet ble utført i disse språkene. Både React og Node.js er utviklingsplattformer for Javascript (Meta Platforms, Inc., 2022) (OpenJS Foundation, 2022). Ingen i prosjektgruppen hadde erfaring i bruken av disse språkene, som gjorde denne oppgaven noe mer krevende og utfordrende.

I tillegg til Javascriptplattformene har vi også benyttet Firebase som er en Google-utviklet programvare designet for å lage web og mobil-applikasjoner (Google Developers, 2022). Det er dette bedriften bruker for å utvikle sine programvarer og er derfor også verktøyet vi skulle bruke. Dette ble gjort for å tilrettelegge for videre utvikling og bedre integrering av løsningen i Styreportalen sitt notearkiv. Firebase er en såkalt «Backend as a service», som altså betyr at alt som foregår på tjenersiden i stor grad kontrolleres av Google, noe vi også måtte ta hensyn til.

2.1.3 Initial løsnings-idé og ressurser

Styreportalen så for seg en løsning der en enkelt kan bruke “Drag and Drop” for å velge hvilke PDF-notesett som skal kategoriseres. Deretter skulle det da skje en automatisk gjennomgang av PDF-ene der notesettet blir kategorisert etter stemme.

Det ble tidlig i prosjektet klart at det i en algoritmebasert løsning, vil kunne oppstå utfordringer med å automatisk kategorisere et notesett korrekt hver gang funksjonen kjøres. Det må derfor settes inn en del tiltak/tilpasninger for å korrigere eventuelle feiltolkninger. Mulige tiltak vi så for oss i begynnelsen var eksempelvis en "Pause"-funksjon eller lignende som kunne kjøres hvis man ikke var helt sikker på om tolkningen av en note er korrekt. For eksempel hvis sikkerheten for riktig kategorisering ligger på <80%, så blir bruker spurt om hva denne siden burde kategoriseres til.

Etter dette så vi for oss at man skal få opp en forenklet oversikt over det kategoriserte notesettet. Her kunne en kjapt se over notesettet og korrigere eventuelle feil. I tillegg måtte også da PDF-en splittes opp i ulike stemmer for å gjøre denne jobben ideell, altså flere mindre PDF-er. Dette var allerede en del av løsningen som styreportalen har muligheter for, så her var det ikke nødvendig å gjøre noe implementering.

2.2 Avgrensninger

Vi har sett det som viktig at en har klare avgrensinger i prosjektet, slik at oppgaveløsningen er effektiv og målrettet. Vi har derfor avgrenset prosjektet vårt til de tema og løsninger som er relevante i forhold til problemstillingen vi fikk fra Styreportalen.

Vi har for eksempel ikke gått inn i problematikken rundt design, altså utseende, for hvordan kategoriseringsverktøyet skal designes for brukerne. Denne delen vil Styreportalen ta seg av når verktøyet skal integreres med de andre løsningene til portalen. Fokuset vårt var derfor å utvikle en løsning som kan ta inn et PDF-notesett og kategorisere dette basert på stemme. Løsningen vår kommer heller ikke til å se særlig på maskinlæring og trening av maskinlærte modeller, men være algoritmebasert. Grunnen til dette er at selv om en får utviklet en modell som kan kategorisere basert på maskinlæring, så vil dette ikke nødvendigvis være nok i seg selv. Vi tror derimot at en kombinasjon av både algoritme og maskinlært modell ville vært en optimal løsning. Men tiden vi har hatt tilgjengelig har ikke gjort det mulig å teste begge disse løsningene sammen. Derfor har vi, etter ønske fra Styreportalen, konsentrert oss om å utvikle den algoritmebaserte løsningen.

2.3 Litteratur om problemstillingen

Det finnes lite litteratur som berører vår problemstilling, altså selve kategoriseringen av notesett basert på stemme. Men det finnes mye faglitteratur om bildegjenkjenning (Schlipsing, et al., 2012), bilde-til-tekstgjenkjenning (Jamshed, et al., 2020) og litteratur ang. notetolkning (Guerrero-Turrubiates, et al., 2014). Vi har derfor benyttet oss av litteratur fra disse fagområdene, da vi fant dette relevant for det vi skal gjøre. Da det meste av litteratur er tilgjengelig online, har Google Scholar vært vår hovedkilde for å finne frem til faglitteratur som omtaler problem relatert til vår problemstilling.

3 DESIGN AV PROSJEKTET

3.1 Forslag til løsning

Styreportalen ønsket en løsning som baserte seg på Google sitt Vision API. Google AutoML (Google Developers, 2022) ble brukt som verktøy for å forbedre resultatet (Prosjekthåndbok møtereferat, Vedlegg). Dette har vi forsøkt å etterkomme i vår løsning og har derfor foreslått en løsning som baserer seg på tjenestene som Google tilbyr. I tillegg var det også et ønske om at løsningen ble skrevet i Node.js og React, slik at det ble enkelt for oppdragsgiver å implementere løsningen i sine eksisterende systemer.

Vi har derfor utarbeidet et forslag til løsning hvor vi baserer oss på Google sitt Vision API for tekstgjenkjenning av PDF noter. Selv med en løsning som kun baserer seg på algoritmer og Googles Vision API kan en oppnå gode resultater. Derfor kom vi da med et forslag til Sømoen om å i første omgang bare sette søkelys på å få laget en løsning som bygger på Vision API og algoritmer. Dersom vi fikk tid, kunne vi også vurdere å teste en AutoML løsning. En kombinasjon av disse to løsningsformene vil nødvendigvis ikke være i konflikt, men heller komplettere hverandre og gi en forsterket løsning.

Selv om vi har landet på et løsningsforslag er det fortsatt ulike måter å gå frem på. Under presenteres de ulike løsningsalternativene vi vurderte før vi satte i gang med utviklingen.

3.1.1 Alternativ løsning 1

Som et første alternativ vurderte vi en løsning der en utvikler et program for tolkning av resultatene fra Vision API-et selv. Dette programmet måtte da klare å finne den relevante delen av den tolkede informasjonen og sammenligne dette med en liste av godkjente stemmer, samt ta høyde for de ulike måtene en kan skrive stemmene på. Til eksempel måtte den gjenkjenne at “2nd cornet” og “Kornett 2” er samme stemme.

3.1.2 Alternativ løsning 2

Et annet alternativ var å lage en helt egen OCR kjerne fra bunnen av og dermed ikke basere oss på Google Vision. Dette ville blitt en billigere løsning i lengden da Google tar betalt for mengden dokumenter som gjenkjennes per måned dersom det er snakk om mer enn 1000 sider (Google Developers, 2022). Naturligvis vil Styreportalen overgå denne grensen da det er snakk om kategoriseringen av flere tusen notesett. Men å gjøre dette ville vært en stor, ekstra, arbeidsbelastning som gruppen vår ikke nødvendigvis hadde klart å håndtere.

3.1.3 Diskusjon av alternativene

Begge alternativ er gode forslag til løsninger. På en side har vi Google sitt Vision API som allerede er ferdig utviklet og klart til bruk. I tillegg er det et høyt nivå av integrasjon mellom API-et og Firebase styreportalen ønsker at vi skal bruke.

På den andre siden har vi alternativet å lage en helt egen OCR. På denne måten sparer vi oss for den kostnaden som påløper dersom man bruker Vision APIet mye, men en mister også muligheten for å bruke en fullt funksjonerende, allerede utviklet, OCR.

Alt dette tatt i betraktning med det faktum at vi skal bruke Firebase som er det styreportalen bruker, så vil det være mest naturlig å velge alternativ 1. På denne måten vil det naturlig bli en høy grad av integrasjon mot Styreportalens allerede eksisterende system og vedlikehold av programmet vil bli lettere.

3.2 Valgt løsning

Mye på grunn til at vi valgte alternativ 1 var at Styreportalen ønsket det, da denne løsningsformen allerede har et visst nivå av integrasjon med googles systemer slik det er bygd opp nå. På denne måten får vi den enkleste løsningen med et akseptabelt resultat innenfor de rammeverk som er satt.

3.3 Prosjektmetodikk

I dette delkapittelet tar vi for oss hvilken utviklingsmetodikk vi brukte under prosessen. Her legger vi fram prosjektplanen som ble brukt under prosjektets utvikling og risikovurderingen vi benyttet oss av underveis.

3.3.1 Utviklingsmetodikk

Vi valgte å benytte oss av iterativ utvikling som utviklingsmetode. Dette mente vi var en passende metode for vårt prosjekt da vi hadde muligheten til å jobbe trinnvis med utviklingen fra først å kunne sende notesettet til Vision APIet til å til slutt kunne laste opp et notesett på en nettside og deretter få en liste over stemmene i notesettet.

Rapporten ble skrevet i samsvar med utviklingen av løsningen vår til oppgaven. Ved å gjøre dette samtidig vil en kunne bedre rapportere og dokumentere fremgangen av prosjektet. Dette vil da også bli en slags form for underveisevaluering da vi oftere må se igjennom rapporten og kontinuerlig forbedre den.

Gruppen vurderte også å ta i bruk Scrum som et hjelpemiddel. Fordelen med Scrum er at det åpner for høy grad av fleksibilitet under utvikling samt en drastisk økning i evne til å håndtere problemer som oppstår underveis (Schwaber, 2022). Men da Scrum er en arbeidsmetodikk beregnet for større grupper er det ikke alle deler som er relevante for oss som gruppe på 2. Derfor har vi heller tatt i bruk kun enkelte elementer til arbeidsmetodikken.

Sprint-utvikling er en sentral del av Scrum-metodikken. Den baserer seg på å dele inn oppgaver som skal gjøres i 1-2 ukers «sprinter», rangert etter grad av viktighet for fremgangen til prosjektet. På slutten kommer en med tilbakemeldinger om hva som gikk bra, hva som ikke gikk bra og hva tiltak en kan gjøre for å fikse dette.

Vi tok i bruk sprint-elementet av metodikken under utviklingen av vår løsning, men baserte sprintene rundt de ukentlige møtene med Sømoen.

3.3.2 Prosjektplan

Vi startet prosjektet med å utvikle en prototype slik at vi enkelt kunne kontrollere at de enkelte funksjonene fungerte slik de var tiltenkt. Her var målet bare å sjekke at ressursene vi skulle ta i bruk fungerte og at vi forstod bruken av dem. Denne lette prototypen gav oss overraskende gode resultater der vi fikk ut JSON-objekter med mye informasjon fra test-PDF-en vår. Dette inkluderte informasjon som forfatter, tittel og, viktigst av alt, stemme. Det neste steget for oss ble derfor å få trukket dette ut som et streng-objekt for videre kategorisering. Googles Vision API viste seg å være greit å jobbe med, godt tilrettelagt for utvikling av programvare som bygger på denne type data, men vi ser at tidsbruk kan vise seg å bli et problem.

3.3.3 Risikovurdering

Tabell 1 - Her har vi gjort en risikovurdering for applikasjonen/programmet vi utvikler. Vi har, ut ifra risikoen, laget planer for å håndtere problemene som kan oppstå (Prosjekthåndbok, Vedlegg).

	Hendelse /Risiko	**Årsak**	**Sannsynlighet**	**Konsekvens**	**Risiko-produkt**	**Tiltak**
1	*Applikasjonen blir ikke klar/ferdig*	*Ikke tilstrekkelig med tid*	*Lav (2)*	*Høy(4)*	*8*	*Planlegge møter med veileder samt Sømoen ofte. Vil opprettholde jevn fremgang.*
2	*Applikasjonen blir ikke tatt i bruk.*	*Ineffektiv og lite brukervennlig*	*Svært lav (1)*	*Høy (4)*	*4*	*Kjøre mange prøvetester/bruker tester.*
3	*Applikasjonen har ikke den forventede ytelsen/ resultater*	*Maskinlærings modellen er ikke trent riktig/algoritme løsningene er ikke gode nok*	*Middels (3)*	*Middels (3)*	*9*	*Teste flere ulike algoritmer/trene forskjellige modeller opp på ulik måte for å finne den beste.*

Tabell 2 - Utregningsmatrise

Sannsynlighet						
	Svært Høy (5)	5	10	15	20	25
	Høy (4)	4	8	12	16	20
	Middels (3)	3	6	9	12	15
	Lav (2)	2	4	6	8	10
	Svært Lav (1)	1	2	3	4	5
		Svært Lav (1)	Lav (2)	Middels (3)	Høy (4)	Svært Høy (5)
	Konsekvens					

For å sikre at arbeidet vårt var i rute brukte vi en risiko vurdering slik som vist i Tabell 1 med forklaring i Tabell 2.

Det som utgjorde den største risikoen ifølge vår risikoanalyse er om applikasjonen ikke har den forventede/nødvendige ytelsen. Dette løste vi ved å teste flest mulig algoritmer for å bedre peile oss inn på den beste løsningen.

Litt mindre risiko var at applikasjonen ikke blir ferdig. Til tross for den relativt høye konsekvensen av dette dersom det skulle skje mente vi at dette utgjorde en mindre risiko på grunn av en ikke altfor stor arbeidsmengde. Med ikke altfor stor arbeidsmengde mener vi at oppgaven vi skal utføre skal være overkommelig i løpet av semesteret vi skriver bacheloroppgave.

Til slutt, med minst risiko, var om applikasjonen ikke ble tatt i bruk. Konsekvensen av dette var naturligvis høy, men sannsynligheten svært lav. Dette er grunnet hvor interessert Styreportalen var i en slik løsning som vi nå har utviklet. I tillegg, pga. den valgte utviklingsformen, vil vi uansett ha et høyt nivå av integrasjon med styreportalens system og dermed tilrettelegge for videre arbeid på programmet.

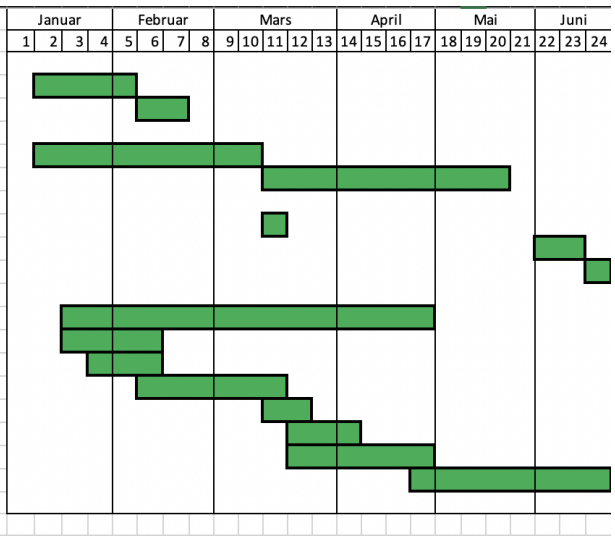
3.4 Evalueringsplan

Evalueringen av prosjektet baserte seg i stor grad på tilbakemeldingene vi fikk underveis fra Sømoen på møtene vi har hatt med ham. Disse møtene ble holdt hver uke, med noen få unntak. I disse møtene kom vi med spørsmål og viste frem det vi hadde så langt. Sømoen kom så med innvendinger, forslag og generelle tips til hvordan vi skulle gå frem med utviklingen av løsningen til oppgaven. Ved å drive evaluering på denne måten fikk vi en pålitelig og god rutine for utviklingen samtidig som programmet kontinuerlig ble evaluert. I tillegg var det naturlig for oss å vurdere resultatene våre ved å sjekke prosentvis hvor mye av notesett som korrekt blir kategorisert. Her tenker vi at vi mot slutten av utviklingen vil mate programmet med 100-1000 notesett av varierende kvalitet for å se hvor mye som

korrekt blir kategorisert. Til sammen vil disse evalueringsmetodene gi oss et godt grunnlag evalueringen av løsningen vi utvikler.

Under i Figur 1 følger en fremdriftsplan for utviklingen av løsningen på vår oppgave. Denne figuren er basert på planen for dette vårsemesteret. Denne planen, sammen med de ukentlige møtene med Sømoen og graden av nøyaktighet under kategorisering, gav et godt grunnlag for underveisevaluering av løsningen.

ID	Status	Oppgave	Planlagt start	Planlagt slutt	Antall timer	Ansvarlig Person
		Undervisning				
1	ok	Forelesninger	11/01/2022	03/02/2022	28	Begge
2	ok	Analyse av tidligere bachelorer	07/02/2022	17/02/2022	30	Begge
		Dokumentering				
3	ok	Forprosjektrapport	11/01/2022	11/03/2022	100	Begge
4	ok	Endelig rapport	12/03/2022	23/05/2022	200	Begge
		Presentering				
5	ok	Forprosjektpresentasjon	12/03/2022	14/03/2022	20	Begge
6		Endelig presentasjon	30/05/2022	08/06/2022	20	Begge
7		EXPO	15/06/2022	15/06/2022	20	Begge
		Utvikling				
8	ok	Utforske mulige løsninger	20/01/2022	29/04/2022	200	Begge
9	ok	Vision API tesing	20/01/2022	10/02/2022	50	Begge
10	ok	Testing av algoritme for tolkning	27/01/2022	10/02/2022	30	Begge
11	ok	Implementering av Firebase	10/02/2022	17/03/2022	100	Begge
12	ok	Implementering av Cloud Funksjoner	17/03/2022	24/03/2022	20	Begge
13	ok	Implementering av test klient	24/03/2022	07/04/2022	40	Begge
14	ok	Videreutvikling av Cloud Funksjoner	24/03/2022	29/04/2022	100	Begge
15		Videreutvikling	29/04/2022	15/06/2022	130	Begge
16		**Ferdig**		15/06/2022		
				sum timer:	1088	



Figur 1 - Gantdiagram v.3 (Prosjekthåndbok, Vedlegg)

.

4 DETALJERT LØSNING

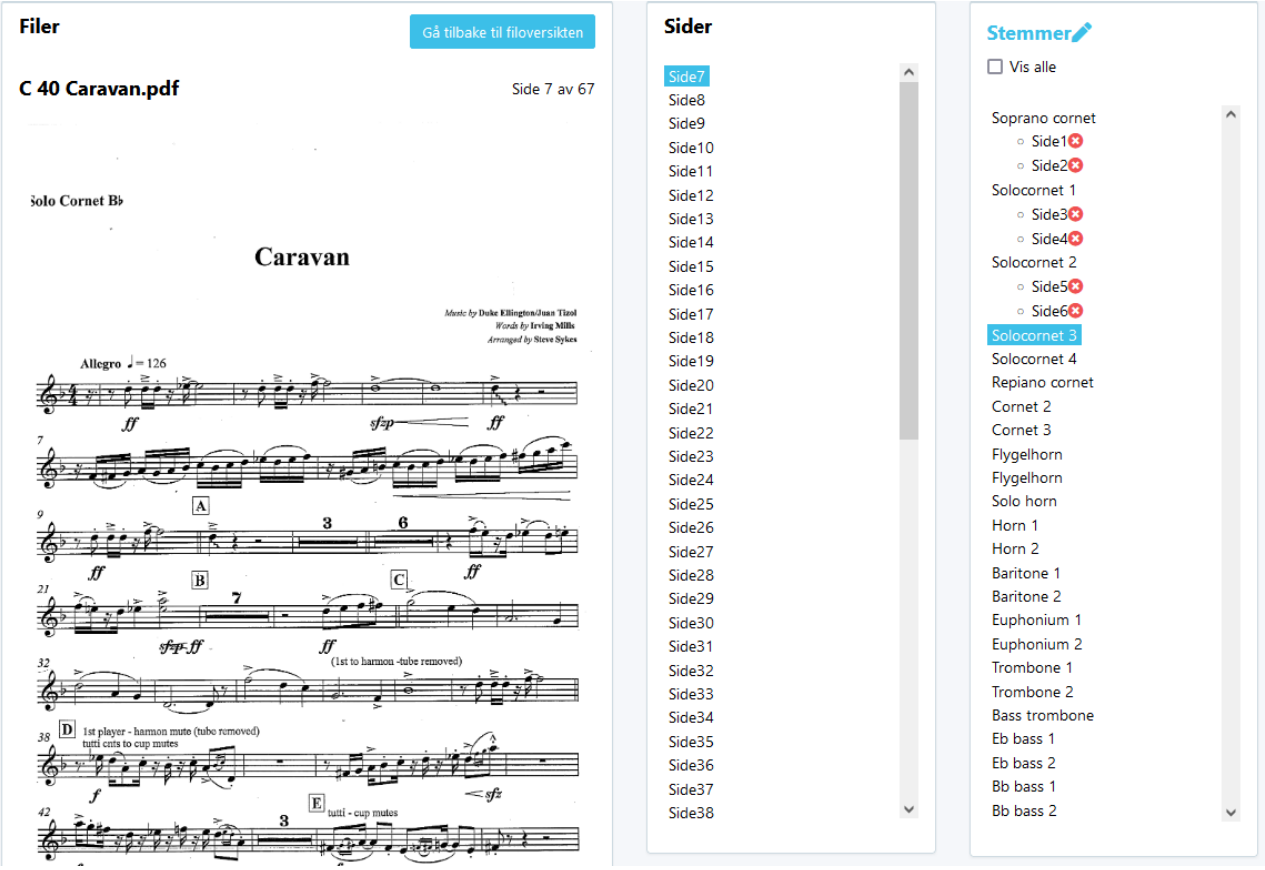
I dette kapittelet vil vi utrede den detaljerte løsningen til oppgaven. Elementer som er mer i dybden av oppgaven vil bli sett nærmere på og de valgte konkrete løsningene vil bli beskrevet.

4.1 Bruk

Som nevnt innledningsvis er løsningen til styreportalen per i dag at brukere laster opp sine noter i PDF-format. Deretter kan en manuelt gjennomgå notesettet side for side og tilordne siden en stemme. For en notearkivar er dette er tidkrevende arbeid som også betyr mye repeterende arbeid.

I Figur 2 ser en hvordan skjemaet ser ut for brukere som velger å kategorisere notene sine etter stemme. De laster opp en PDF i portalen som tar de til denne siden. Styreportalen vil automatisk dele notesettet opp i sider slik at bruker kan se en forhåndsvisning av de ulike sidene. Her går da bruker manuelt igjennom, ser hvilken stemme de ulike sidene har og tagger dem korrekt. Da vil styreportalen sortere dem etter den stemmen de er tagget som.

Styreportalen ønsker derimot at denne prosessen skal automatiseres. I en slik løsning så vil bruker bare laste opp en PDF av et notesett, notesettet vil så bli gjennomgått og deretter kategorisert/tagget basert på stemme helt automatisk. Bruker må da naturligvis vente litt på at programmet får gjort alt det trenger å gjøre, men etter en liten stund vil de få opp varsel om at notesettet er blitt kategorisert. Det var allerede på et tidlig tidspunkt i prosjektet tydelig at en slik algoritmebasert løsning ville kunne får problemer med å utføre denne prosessen korrekt hver eneste gang. Disse kategoriseringsproblemene er det flere grunner til. Hovedutfordringen er imidlertid at det ikke finnes noe standard oppsett for hvordan et notesett skal være bygd opp. Stemmer skrives på svært forskjellige måter og noter kan være ment for flere stemmer samtidig.



Figur 2 - Dagens løsning for stemmeinndeling i Styreportalen

Arbeidet vårt har vært konsentrert rundt selve kategoriseringsprosessen, da målet vårt hele tiden har vært å utvikle et system som kan lese gjennom et notesett lagret som en PDF, og returnere informasjon om hvilken stemme hver side tilhører. Systemet skal implementeres i samme applikasjonsmiljø som Styreportalen allerede bruker, slik at det skal være så enkelt som mulig for Styreportalen å bruke det inn i sine allerede eksisterende systemer, samt gjøre det enkelt å videreutvikle.

4.2 Programflyt

Før vi gikk i gang med den mer detaljerte løsningen var det viktig å ha klart for oss hva vår løsning faktisk skulle gjøre.

Løsningen vår skulle altså ha følgende programflyt:

- Ta imot en PDF-fil som input
- Sende PDF filen til Google Vision for tekstuthenting
- Vha. algoritmer gjenkjenne stemmen på de ulike sidene.
- Returnere stemmeinndeling

Et annet viktig moment var at vi skulle implementere løsningen i Firebase da dette ville gjøre det lettere for Styreportalen å integrere dette inn i allerede eksisterende systemer som allerede benytter Firebase som plattform.

4.3 implementering av algoritmer

4.3.1 Implementering av Google Vision API

Det første vi måtte gjøre var å implementere en metode som gjorde det mulig å la en applikasjon sende en PDF til Vision og få et JSON objekt med resultater i retur. Google har litt eksempelkode (Google, 2022) som viser hvordan kalle API-et i forskjellige programmeringsspråk, deriblant Node.js som React er basert på. For å kunne bruke Vision API-et må en autentisere seg, og den foretrukne metoden for dette er å bruke såkalte service accounts (Google, 2022). For å unngå at nøklene til disse kontoene kan komme på avveie, ønsket vi å implementere all logikken for autentisering og kommunikasjon med Vision API-et på tjenersiden av applikasjonen. Styreportalen er bygget på Firebase som er en såkalt «Backend as a Service». Det betyr at det ikke er mulighet for å skrive kode direkte på tjenersiden. Firebase tilbyr i stedet Cloud Functions (Firebase, 2022) som lar utviklere kjøre tjenerside kode. Vi valgte derfor å benytte oss av disse, for å sikre at løsningen vår kan brukes av Styreportalen.

Cloud Functions kan enten utløses direkte ved http forespørsler, eller de kan bli utløst automatisk som en respons på en hendelse i Firebase som eksempelvis at en fil lastes opp (Google, 2022). Under utviklingen ble begge metoder vurdert og testet, og vi oppdaget ganske tidlig i prosjektet at direkte utløsning med http-forespørsler var det foretrukne alternativet for vårt formål. Grunnen til dette er at selv om hendelsesutløsning kan utføre hele arbeidet med å tilordne hver side til en stemme, så er det ingen enkel måte å sende denne informasjonen direkte tilbake til klienten igjen. Dette er fordi at funksjonen ikke har noen måte for å finne ut hvilken klient som lastet opp filen som utløste funksjonen. Vi hadde et ønske om at når resultatet var ferdig så skulle klienten få umiddelbar respons, og da var http-forespørsler en god løsning da vi kan sende resultat tilbake igjen til klienten med http-responsen som svarer forespørselen.

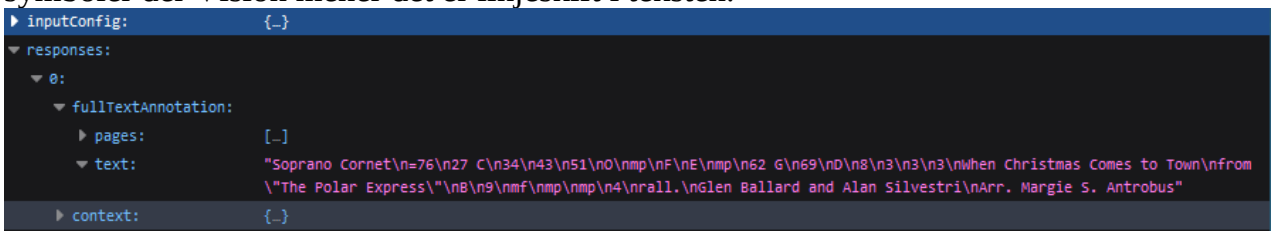
I vår endelige implementering av Vision API-et valgte vi derfor en løsning der vi sender en http-forespørsel som inneholder informasjon om plasseringen til filen vi ønsker at Vision skal uthente tekst fra. Funksjonen bruker så denne informasjonen til å utforme en forespørsel (Google, 2021) om tekstuthenting og sender denne forespørselen til Vision API-et. Når Vision har analysert filen så lagres responsen på en plassering spesifisert i forespørselen (Google, 2022). Denne plasseringen sender vi så tilbake http-responsen, slik at klienten kan be om tilordning av stemmer basert på resultatet av tekstuthentingen.

4.3.2 Google Vision API resultat

Når Vision har analysert filen ferdig lagres resultatet på den spesifiserte plasseringen (Google, 2022). Innbakt i systemet er det også mulig å bestemme hvor mange sider fra PDF filen som er med i hver resultatfil. Dersom det er flere sider i PDF filen vil det lagres flere resultat filer. Hvor mange sider som er med i hvert resultat kontrolleres av variabelen batchSize (Google, 2022). For vår bruk var det mest hensiktsmessig å lagre alt i en fil, da vi uansett var nødt til å behandle all informasjon. Det var derfor naturlig å sette batchSize til en så høy verdi som mulig, siden verdien spesifiserer det største antallet filer som er

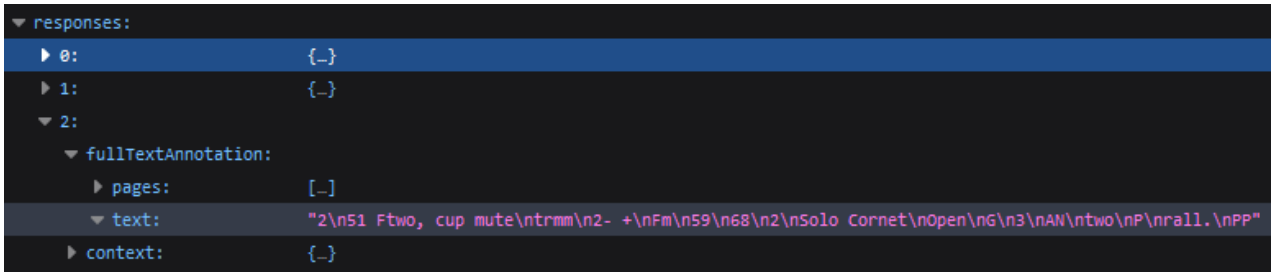
inkludert i en respons. Setter vi verdien til 100 så vil resultatet til alle notesett på under 100 sider lagres som en enkelt fil.

Resultatet fra Vision-analysen lagres som en JSON fil som er utformet slik at all informasjonen for hver side lagres som en respons (Google, 2021). Hver respons har igjen en kontekst felt som inneholder informasjon om hvilket sidetall dette er, og et felt som heter *fullTextAnnotation* (Google, 2021). Under *fullTextAnnotation* er det to nye felter igjen. Først kommer *pages*, som bare inneholder et element når Vision blir kalt med den konfigurasjonen vi bruker. Deretter kommer *text*, som inneholder all gjenkjent tekst på siden som en lang streng hvor hver paragraf er splittet med «\n» (newline). I tillegg er det også «newline» symboler der Vision mener det er linjeskift i teksten.



Figur 3 - Text felt fra første side av When Christmas Comes to Town, Arrangert for brassband av Margie S. Antrobus.

Et eksempel på slikt tekstfelt er vist i Figur 3. Her ser man at stemmen, som i dette tilfellet er *Soprano Cornet*, kommer tidlig i tekststrengen og kan brukes til å identifisere stemmen som hører til siden. Dessverre er ikke alltid stemmen først i strengen slik som vist i Figur 4. Der kan en se at stemmen ikke dukker opp før i midten av strengen. Vi er derfor nødt til søke gjennom hele strengen for å finne stemmen. Dette kan bli problematisk dersom eksempelvis tittelen til stykket inneholder en stemme, noe som ikke er utenkelig da det finnes stykker hvor et instrument spiller solo hele stykket. Eksempel på dette kan være Napoli av Herman Bellstedt arrangert av Geoffrey Brand (Just Music UK, 2022). Her er *Cornet Solo* alltid skrevet på tittelsiden og vil da kunne bli gjenkjent som instrumentstemmen selv om den ikke er det.



Figur 4 - When Comes to Town Solo Cornet side 2

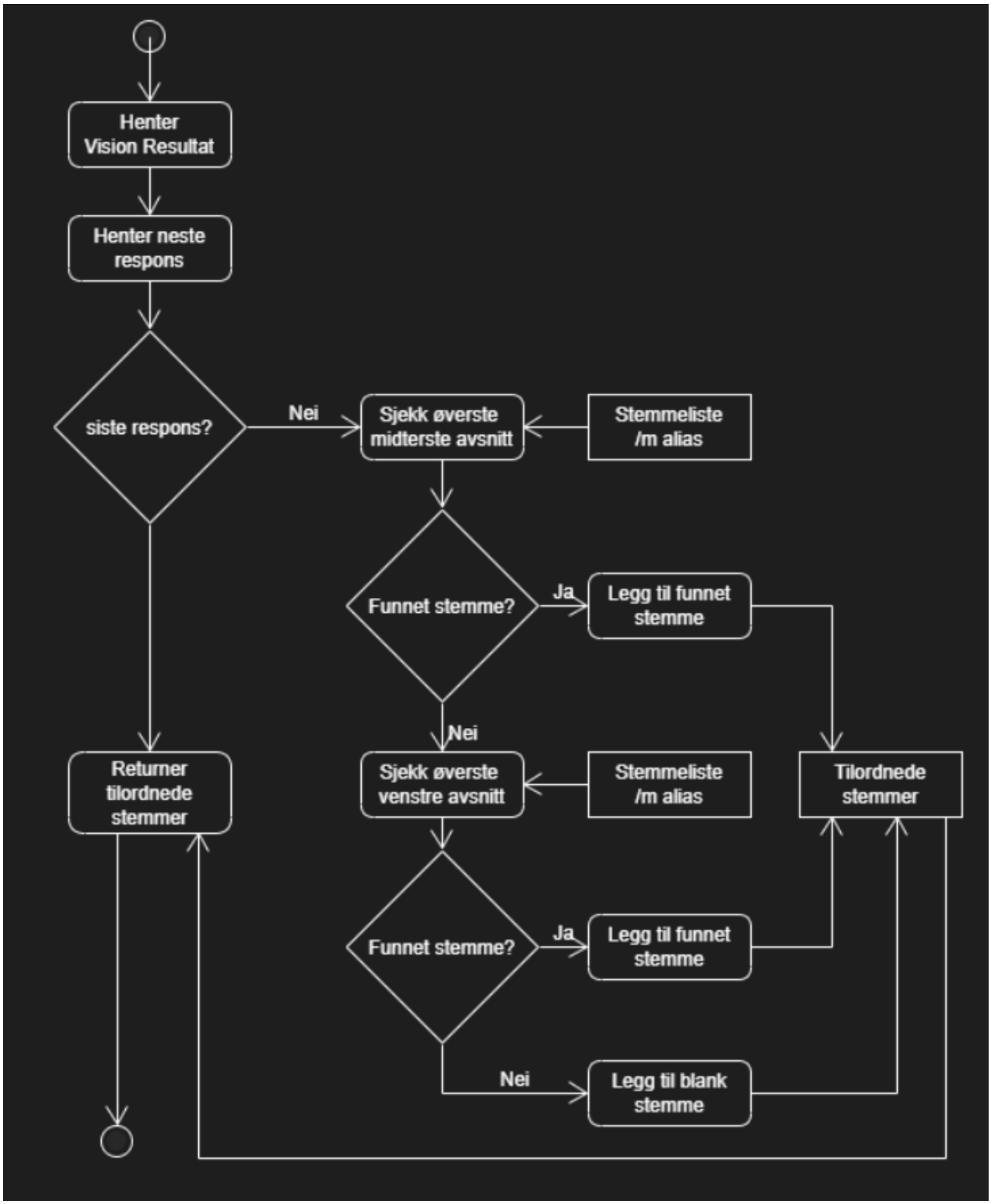
Det ble etter hvert klart for oss at feltet *text* ikke har nok informasjon til at vi kan bruke det direkte for stemme-gjenkjenning, men at vi kunne bruke den for å sjekke om en side er en del av et partitur eller ikke. Det kommer vi tilbake til senere.

Siden *text* feltet ikke hadde nok informasjon, bestemte vi oss for å se nærmere på *pages* i stedet. Pages inneholder en liste av *blocks* som igjen inneholder en liste av *paragraphs*. Hvert *paragraph*-felt inneholder en liste av ord og informasjon om avgrensningsboksen definert av fire relative posisjoner på siden. Ved hjelp av disse avgrensningsboksene kunne vi sortere *paragraph*-feltene, alt etter hvor på siden de befinner seg. Denne informasjonen

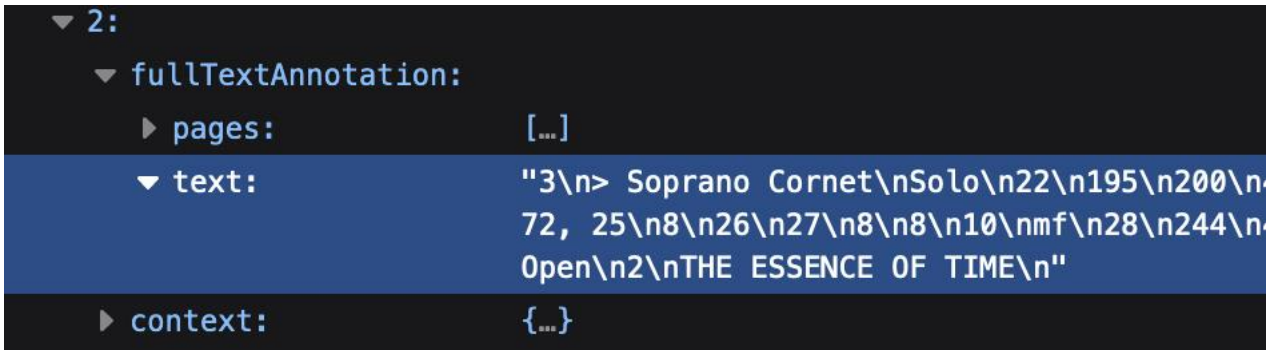
kan for eksempel brukes for å finne feltet som er øverst på siden. Dette er interessant ettersom stemmeinformasjonen stort sett alltid er øverst til venstre på første side, og øverst i midten på de påfølgende sidene med samme stemme. Det er derimot ikke noe felt under *paragraph* som inneholder all tekst i paragrafen. For å hente ut teksten i en *paragraf* er vi nødt til å gå gjennom alle ordene under paragrafen og deretter alle symbolene i hvert ord for å bygge opp hva som står i paragrafen. Her må vi også dra nytte av et mulig felt under symbol som kalt *property* som igjen inneholder *detectedBreak* som har feltet *type*. Dette feltet forteller oss om symbolet er etterfulgt av mellomrom, newline eller lignende. Ved hjelp av symbolene og *detectedBreak* kan vi hente ut teksten i en spesifikk paragraf.

4.3.3 Algoritme for tolking av Vision resultat

Når relevante avsnitt er hentet ut, må de prosesseres og basert på informasjon må gjeldende side tildeles en stemme. Der er en mulighet å bare returnere øverste venstre avsnitt og satse på at det er rett stemme, men dette vil ikke være noen god løsning. I stedet valgte vi å gå for en løsning der vi har en liste over stemmer som vi sammenligner hvert avsnitt med. Dersom det avsnittet finnes i listen over stemmer, er det riktig stemme. Men dessverre er det slik at stemmene ikke alltid er like mellom stykker. I tillegg kan også samme stemme skrives på forskjellige måter. For å løse dette problemet laget vi en liste som inneholdt de vanligste stemmene som er brukt i brass band og janitsjar korps. Og for hver stemme er det igjen lagret en rekke med aliaser for de ulike måtene å skrive denne stemmen på. På den måten kan vi ta høyde for flere forskjellige måter å skrive samme stemme på. For å unngå problemer med forskjeller på store og små bokstaver lagrer vi alle aliasene med kun små bokstaver og casefolder avsnittets innhold før det sjekkes. Et aktivitetsdiagram som for denne prosessen er vist i Figur 5.



Figur 5 - Aktivitetsdiagram for tilordning av stemmer



Figur 6 - JSON med feiltolkning

Dessverre klarer ikke alltid Vision å tolke PDF filene korrekt. I Figur 6 ser vi et eksempel på at Vision ikke alltid klarer å tolke rett. Korrekt stemme på denne siden er «*Eb Soprano Cornet*», men Vision har tolket det som «*> Soprano Cornet*». Dersom vi bare sjekker om avsnittet er helt lik et alias, ville ikke denne siden ha returnert et gyldig resultat. Det er derfor nødvendig å sammenligne avsnittet og aliasene på en måte slik at de kan regnes som like selv om de ikke er helt identiske. For å gjøre dette trenger vi en algoritme for strengsammenligning og det beste for oss var derfor å benytte oss av en allerede kjent algoritme.

Vi testet først med å bruke Levenshtein distanse (Levenshtein, 1966) for å sammenligne strenger. Levenshtein distanse er en forholdsvis enkel sammenligningsalgoritme. Denne algoritmen teller antall endringer som må gjøres for at en gitt streng skal bli lik en annen. Eksempelvis vil *eb soprano cornet* og *> soprano cornet* ha en Levenshtein distanse på 2. Et bytte fra > til e, og en innsetting av b. Vår test-implementering fungerer ved å sammenligne en paragraf med alle aliasene i stemmelisten, for å så velge den stemmen med alias der Levenshtein distanse er lavest. Dette løste det primære problemet med feiltolkingen, men viste seg likevel ikke å være en løsning som fungerte i alle tilfeller.

Under testing kom problemet med korte ord som ofte har en Levenshtein distanse på 3 eller mindre mot stemmen obo. Dette gjorde at mange sider ofte blir tolket som obo selv om obo ikke er en del av besetningen. For å løse dette problemet kunne en for eksempel sørge for at distansen ikke er større enn lengden på halvparten av strengen. Dessverre vil det da fortsatt være problemer med aliaser som er ganske nær hverandre i Levenshtein distanse. Et eksempel på dette er side 2 på en flygelhorn-stemme som blir tolket som *flugel – 2*. Denne paragrafen har en Levenshtein distanse på 4 mot aliaset flugel, men også en Levenshtein distanse på 2 mot aliaset *flute – 2*. Ut ifra dette kommer det fram at vi må ta høyde for rekkefølgen på bokstavene. Heldigvis finnes det en sammenligningsalgoritme som gjør akkurat dette.

4.3.4 Jaro-Winkler

En annen algoritme som også tar hensyn til rekkefølgen til bokstavene, er Jaro-Winkler. Vi endte opp med å bruke denne algoritmen da den gir høyere score dersom strengene har sammenfallende prefiks av en viss lengde (Winkler, 1990). Ved å bruke denne algoritmen så forsvinner omtrent alle problemene vi har med feiltolkinger, og de som nå og da dukker opp kan enkelt løses ved å legge inn nye alias. De dukker stort sett opp i slagverksseksjonen

sine stemmer da stemmene der kan skrives på mange flere måter da de har store mengder med instrumenter som kan brukes.

4.3.5 Partitur

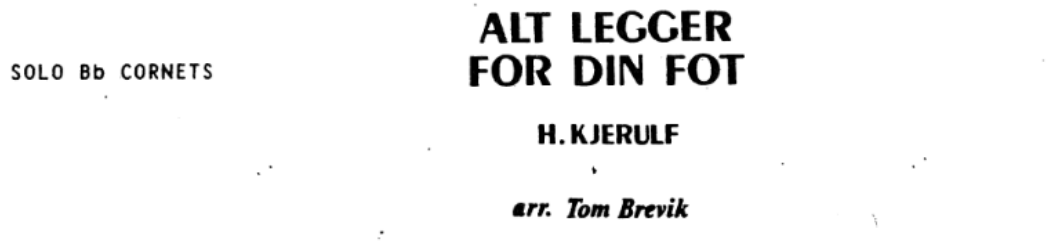
Selv om stemmene nå begynner å tolkes rett, så er det fremdeles en utfordring å få til en riktig kategorisering av partitur (Nergaard, 2020). Et partitur er notene en dirigent for et korps leser. Det viser en oversikt over alle stemmer for dirigenten slik at samspill er lettere for musikantene. Partituret kan av og til skrives som *score,* eller *conductor* og vi kan legge disse inn som alias, men det er ikke alltid at det står skrevet direkte at det er partitur. Det som derimot nesten alltid er likt, er at det før hver notelinje er skrevet hvilken stemme dette tilhører. Vi brukte derfor en funksjon som sjekket gjennom *text* feltet nevnt i 4.3.2 og telte hvor mange linjer som matchet med stemmelisten vår. Dersom over 10 stemmer matcher, så kan vi med stor sannsynlighet anta at det dreier seg om en partiturside.

4.3.6 Utvidet område

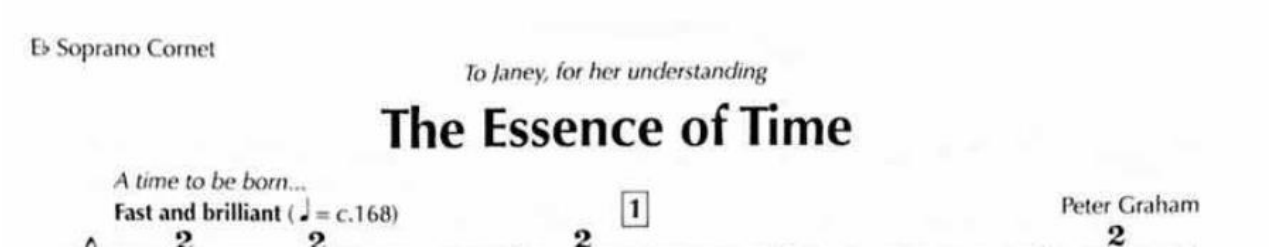
I tillegg til stemmen ønsker vi ideelt sett også å få ut tittel på stykket, samt komponist og arrangør. Dette er i utgangspunktet ikke en del av oppgaven som vi har blitt gitt, men er noe som er veldig relevant for videre bruk av programmet da målet er så mye automasjon som mulig.

Som det ble sagt tidligere er stemmen til et stykke typisk notert øverst i venstre hjørne. På samme måte er tittel på stykket som regel skreve i senter av siden helt øverst og komponist med arrangør notert til høyre helt øverst på siden. Et eksempel på en slik note er vist i Figur 7. Men i disse tilfellene er det ikke bare for oss å lage en ordliste over godkjente ord som programmet som skal lete etter her. Tittel, komponist og arrangør varier veldig fra stykke til stykke. Dermed må vi håndtere dette problemet på en annen måte enn tidligere.

Figur 7 - Standard oppsett på stemme, tittel, komponist og arrangør



Figur 8 - Eksempel på avvik fra standard oppsett



Figur 9 - Eksempel på avvik i tittel

I Figur 8 og Figur 9 ser vi eksempler på at tittelen ikke alltid er øverst i midten på siden. Det vises også at komponist og arrangør heller ikke alltid er skrevet til høyre på siden.

Det som derimot ofte er felles med titler er at skriften som regel har størst størrelse på siden. Dette kan dras nytte av for å identifisere tittelen. En kan hente ut tittelen ved å hente teksten med størst skrift fra siden hver gang stemmen på siden er ulik stemmen på forrige side. Deretter kan man lage en liste med alle potensielle titler og velge det elementer som forekommer flest ganger som den antakelige tittelen.
Arrangøren kan som regel identifiseres med et prefiks på formen *Arr:, Arranger,* eller *Arrangør*. Dette kan vi dra nytte av ved å lete gjennom feltet *text* i responsen for hver side, og ved hjelp av regex hente ut alle linjer som begynner med «arr».
Komponisten er det derimot ikke noe så åpenbart unikt med. Ettersom komponisten sitt navn som oftest er skrevet på høyre side av noten, og alltid over eventuell arrangør, mener vi det er mest hensiktsmessig å anta at linjen med tekst øverst til høyre på siden er komponisten.

4.4 Applikasjon for live-test

Etter å ha utviklet fungerende cloud funksjoner, var neste steg å lage en app i Firebase for å teste at systemet fungerer. Applikasjonen er en enkelt side app hvor man kan laste opp en enkelt PDF fil, og få i retur en liste over stemmer i rekkefølge sammen med antatt tittel og komponist. For å teste at løsningen vår kan brukes i miljøet som styreportalen bruker, kodet vi demoapplikasjonen i React.

4.5 Database for stemmer og notesett

Løsningen vår benytter en database over godkjente stemmer. I denne databasen legges de ulike stemmene inn med kjente aliaser knyttet til dem. Det er ikke alle stemmer vi vil klare å få dette til å fungere for, da det finnes mange ulike måter å skrive en stemme på. Eksempelvis kan «kornett 1» skrives som «kornett 1», «1st. Kornett» og «1 kornett». Dersom gruppen vår skulle skrevet alle forskjellige versjoner manuelt, ville det skapt mye arbeid for bare to personer. Men hva om brukerne gjør denne jobben? Det er fult mulig å utvikle en funksjon som gjør at brukere manuelt kan legge inn en stemme dersom den ikke dukker opp i databasen. Da vil programmet vårt etter hvert få et mye større repertoar av stemmer den gjenkjenner, og ytelsen til løsningen vil naturlig øke med bruk. Men dersom denne funksjonaliteten legges til som en integrert del av løsningen, må en også ta høyde for om brukere faktisk legger inn reelle forslag. Dersom brukerne kan legge inn hva de vil, hvor de vil, kan det raskt bety dårligere resultater.

Et annet viktig moment er at vi kan effektivisere denne prosessen ytterligere dersom dette stykket ligger ferdig kategorisert i databasen. For hvis dette stykket allerede ligger ferdig kategorisert i databasen kan vi raskt hente ut kategoriseringen fra databasen og angi den på filen brukeren har lastet opp. Det er viktig å presisere at det kun er kategoriseringen angitt til notesettet som vil bli hentet ut fra databasen og ikke selve filen som er ferdig kategorisert. For å få dette til å fungere er det imidlertid viktig at en er 100% sikker på at filen bruker laster opp faktisk stemmer overens med det som allerede ligger inne i databasen til styreportalen. Sømoen stilte seg skeptisk til forslaget, og vi endte opp med å la være å prøve å implentere denne funksjonaliteten inn i løsningen da det er vanskelig å oppnå en 100% grad av sikkerhet.

4.6 Optimering av løsning

Hovedmålsettingen med utviklingen av dette katalogiseringsverktøyet for PDF filer er å redusere mengden tid det tar å kategorisere notesett. Men underveis i arbeidet vårt har vi sett at at løsningen vår bruker en del tid på å kategorisere notesettene vi legger inn. Det er i hovedsak to grunner til dette.

Årsak nr. 1 er hvordan Vision API-et til google er bygget opp. Dette er det ikke så mye vi får gjort noe med. Vi trodde til å begynne med at tidsbruken skyltes at den brukte mye tid på å komme igjennom notelinjene som var i notesettene. Vi prøvde derfor å gi den kun PDF-filer som inneholdt den øverste delen av notearket, da det er her instrumentstemmen typisk er skrevet inn. Men API-et brukte like lang tid på kategoriseringen som før. Med andre ord har vi ikke noen direkte måte å redusere til Vision APIet..

Årsak nr. 2, er tidsbruken som tolkingen av resultatene tar. Denne tidsbruken er mye mindre enn tidsbruken Vision API-et bruker, men er like fullt et område som vi har sett på. I vårt tilfelle ble dette en del bedre når vi brukte ferdige algoritmer for streng sammenligning, i stedet for å implementere dem selv. Eksempelvis viste Jaro-Winkler seg for å være mye kjappere enn Leveshtein, da den ikke beregner samme avstand på nytt dersom den feiler.

Vi ser at det nå tar omtrent like lang tid å automatisk kategorisere et enkelt notesett som om en skulle ha gjort dette manuelt. Men vi har også funnet ut at vi kan kjøre flere instanser av cloud funksjonene samtidig. Det betyr at dersom vi kan lage en applikasjon som lar oss laste opp flere noter av gangen, kan vi tolke disse parallelt og likevel ikke bruke mer tid enn om vi bare hadde tolket en enkelt PDF-fil. På denne måten kan tiden kategoriseringen tar, kraftig reduseres.

Kildekoden er lagt på GitHub og beskrivelse av hvordan sette opp løsningen i et Firebase og Google Cloud Platform prosjekt er skrevet i systemdokumentasjonen (Systemdokumentasjon, vedlegg)

5 RESULTATER

I dette kapittelet tas det en nærmere titt på resultatene, samtidig som det gis en evaluering av disse, basert på evalueringsmetodene som ble uttrykt i kapittel 3.

5.1 Evalueringsmetode

Som nevnt tidligere i kapittel 3 har vi i hovedsak brukt tre former for evaluering av programmet vi har utviklet, der to av dem er i form av underveisevaluering. Nummer én er underveisevaluering, gjennomført i samarbeid med Sømoen i form av ukentlige møter. Gjennom disse møtene fikk vi klare tilbakemeldinger og innspill om hva han tenkte kunne være neste steg i programutviklingen. I tillegg har vi også fått innspill og ideer for videre utvikling av programmet, utover det som var det egentlige hovedformålet med oppgaven, nemlig kategorisering av noter basert på stemmer.

Den andre formen for evaluering er utviklingsplanen vår, som vi har vist i form av et Gannt-diagram (figur 1). Denne planen er basert på milepælene vi skulle gjennom i løpet av semesteret.

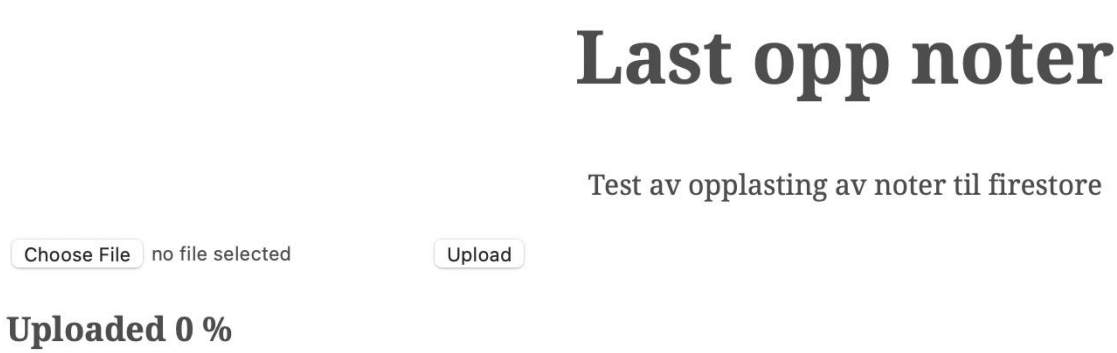
Den siste evalueringsformen av det endelige produktet ble gjort statistisk i form av reelle test-kjøringer. I disse testene ble det kjørt fra 10-68 notesett igjennom programmet. Resultatene fra testkjøringene viste hvordan kategoriseringsprogrammet faktisk presterte da det ble matet med mange forskjellige notesett av varierende kvalitet.

5.2 Evalueringsresultat/Prosjektgjennomføring

Møtene med Sømoen har blitt gjennomført hver uke siden januar, med noen få unntak. Gjennom disse møtene har vi fått verdifull informasjon og gode innspill fra Sømoen. Sømoen utrykte i all hovedsak positivitet til arbeidet som ble gjennomført samtidig som han kom med gode innspill for hva som kunne være neste steg i utviklingen av programmet. Dette kunne inkludere alt fra enkle ting som forslag til hvordan vi kunne optimere deler av algoritmen for kategorisering, til mer omfattende oppgaver som å få alt til å fungere i et Firebase-prosjekt som alle kunne teste på nett. Disse ukentlige møtene og den gode oppfølgingen fra Sømoens side har ført til at vi kom tidlig i gang med prosjektet. På denne måten har vi fått en jevn og trygg iterativ utviklingsprosess.

Gjennomføringsplanen som ble utarbeid tidlig i semesteret har vi derimot ikke helt klart å følge slik den opprinnelig var satt opp. Gjennomføringsplanen ble lagd som et hjelpemiddel for at utviklingsarbeidet skulle ha en så jevn og god progresjon som mulig. Problemer med å få en tidlig versjon av programmet opp på nett og innleveringene som har kommet underveis i de andre fagene har imidlertid medført at det i perioder har vært utfordrende å

følge planen. Figur 10 viser en skjermdump fra en tidlig versjon av test klienten vår, hvor en kan teste hvilken inndeling løsningen vår gir til et enkelt notesett i en pdf-fil.



Figur 10 - Denne figuren viser hvordan test-siden ser ut der vi tester tolkningen av notesett. Merk at dette er en test-side og ikke det endelige produktet for hvordan det vil se ut på Styreportalen.no

Vi mener det ikke skal være særlig store problemer med brukervennlighet da løsningen vår er utviklet med et modulært design i tankene slik at den er lett å koble sammen med et allerede eksisterende system. Det skal ikke være vanskeligere eller mer å gjøre for en notearkivar ved bruk av det nye systemet. Da det er Styreportalen som skal stå for designet av utseendet til GUI-et til Styreportalen.no er det naturlig at det er de som står for den faktiske implementasjonen av løsningen og å få det koblet sammen med resten av deres systemer. Vi har derfor lagt vekt på å lage løsningen vår så modulær som mulig, slik at brukervennlighet skal være enkelt å få til når den endelige løsningen skal innlemmes i de allerede eksisterende systemene til Styreportalen.

5.3 Prosjektresultat

For å evaluere ytelsen til løsningen vi har utviklet har vi kjørt to tester på programmet. I den første testen matet vi programmet med ti notesett i det mest vanlige formatet, A4, med grei kvalitet på tekst og oppsett. Vi la også inn to notesett i et annet format blant de ti notesettene slik at vi kunne sammenligne resultatene basert på hvor stor andel av sidene som faktisk ble korrekt kategorisert. Vi ser her på prosentandelen som er korrekt kategorisert samt hvordan dette henger sammen med hvilke noter programmet har blitt matet med. Basert på de notesettene som ikke ble riktig kategorisert gjorde vi ytterligere justeringer for å forbedre programmet. Det er viktig å presisere at listen programmet vårt bruker over godkjente stemmer er basert på brassband. Dermed er ikke janitsjar, altså tre-blås i betraktning her. Testene som ble kjørt inneholdt derfor kun brassband-noter, da det var dette vi hadde tilgjengelig.

Etter å ha matet programmet med disse ti notesettene var resultatet at 75% av sidene ble kategorisert korrekt. Selv om dette er et relativt høyt tall, var vi likevel ikke helt der vi ønsket å være. Ved hjelp av ulike tilpasninger og justeringer mente vi det burde være mulig å få en

korrekt kategorisering på opp mot 90%. Det var derfor svært viktig å analysere årsakene til hvorfor bare 75% av sidene ble riktig kategorisert. Og ser vi nærmere på bakgrunnen for disse resultatene, så ser vi at det i hovedsak er de to notesettene av et annet format som vi la inn bevisst som er årsaken til den «lave» uttellingen der nesten alle kategoriseringer har feilet som vist i Figur 11.

	funnet	total	kommentar	Tittel	Komponist		
Alt legger for din fot	28	28	mye doble stemmesett her, tok bar første stemme	bom	bom	treffprosent	75 %
Amazing grace	28	35	Bommet på partitur. Percussion ble perkusjon 1	treff	bom		
A night in Tunisia	29	30	Bommet på solo trombone, ikke lagt inn. Dobbletsidig	bom	bom	uten outliers	89 %
As the Deer	30	30	Flygehorn har samme stemme som 2. kornett.	treff	bom		
Barnard Castle	1	29	Dobbletsidig. Haukås logo oppe til venstre skapte problemer.	bom	bom		
Big Spender	23	24	Bommet på Mallets. Ikke lagt til i parts listen	treff	bom		
Bring back that Leroy Brown	48	51	Bommet på 2 slagverk stemmer, en Bb bass ble Eb bass, solo cornet ikke splittet.	treff	bom		
Caravan	53	67	Bommet på en del slagverk. Mangler side 2 og 3 med tuba.	treff	bom		
Christmas song	0	28	Bommet på alle, antakeligvis grunnet lang undertittel som strekker seg over hele siden	treff	bom		
Capriccio Brilliante	22	28	Bommet på andre side solo horn, sto ikke noen stemme der. Bommet på trommer og bass grunnet alias mangel				

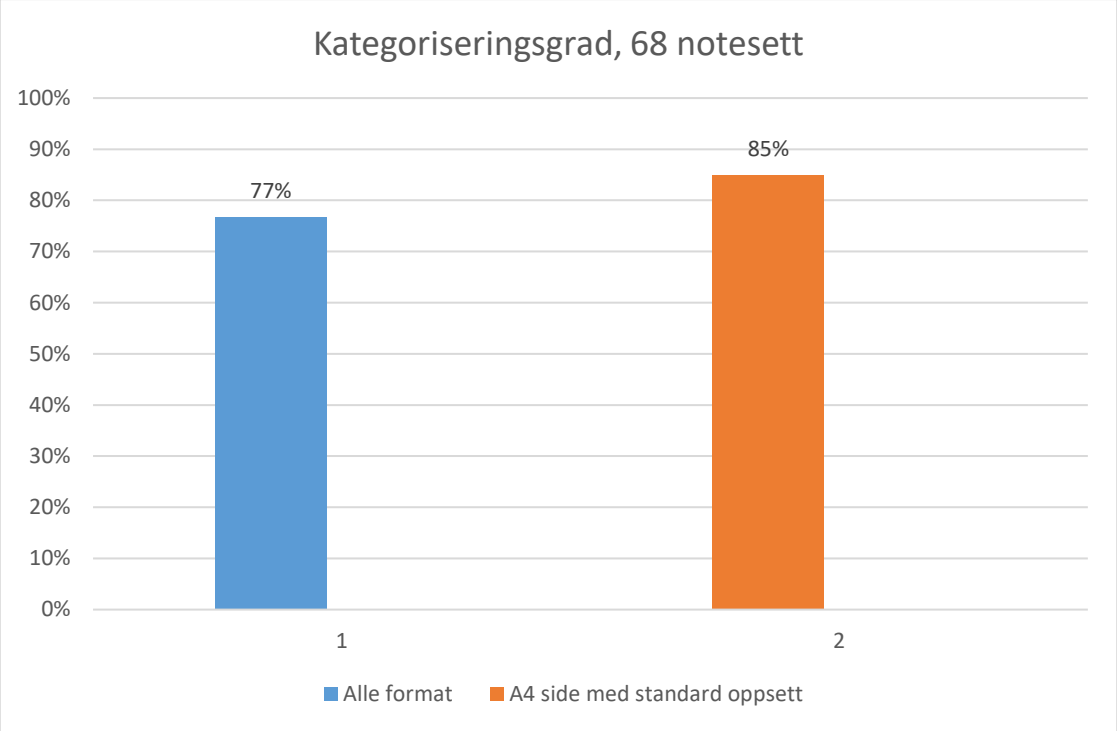
Figur 11 - Denne figuren viser en oversikt over hva notesett som er tatt med i test 1, samt resultatene av kategoriseringsforsøket på dem med tilhørende kommentarer

Det ene av de to notesettene (Barnard Castle) var et notesett bestående av tosidige stemmer. Dette har vi ikke tatt høyde for i algoritmen som brukes for kategorisering. Da kategoriseringsverktøyet går etter prosentvis beliggenhet på siden for å søke etter korrekt stemme, vil den ikke plukke opp stemmen siden denne informasjonen ligger på «feil» plass. Dette er noe som kan tas høyde for ved å lage en egen algoritme for tilfeller der det er tosidige notesett som blir sendt inn. Dette er imidlertid ikke noe vi har implementert per i dag.

Det andre tilfellet vi har der alle kategoriseringsforsøk feiler er notesettet «Christmas Song» der undertittelen til stykket er så lang at den strekker seg hele veien bort til det algoritmen ser på som «øvre venstre side». Dette fører til at den blir feilaktig tatt med som en del av stemmekategoriseringen.

Hvis vi da ser bort fra de to notesettene som ble feilaktig kategorisert blir treffprosenten derimot mye høyere. Hele 89% av notene har blitt kategorisert korrekt, noe vi anser som et godt resultat.

Som en test nummer to kjørte vi igjennom alle notene vi hadde tilgjengelig, totalt 68 notesett. Disse 68 notesettene inkluderer da også de ti notesettene vi kategoriserte tidligere i den første testen. I denne andre testkjøringen ble det altså ikke tatt hensyn til hverken format eller kvaliteten til notesettene.



Figur 12 - Her vises hvor mange prosent korrekt kategorisering som er oppnådd under test 2. Figuren viser to søyler, med og uten standardoppsett. Alle formater inkluderer marsjnoter, og dobbletsidige noter..

I figur 12 ser vi prosentandelen av korrekt kategoriserte noter oppnådd under den andre testen. Selv med noter på ikke-standard format ble det oppnådd 77% korrekt kategorisering. Uten ekstremaler, vist ved søyle 2 i figur 12, ser vi at vi får en 85% korrekt kategorisering.

I den første testen så vi en høyere prosentandel for korrekt kategorisering. Dette naturlig da test 1 til en viss grad ble skreddersydd for systemet. Ser vi nærmere på dataene over hvilke notesett som ble testet under test 2, ser vi at årsaken ligger i notesettene nr. 46 og 47 (Resultater, Vedlegg). I disse to notesettene var det henholdsvis bare 18 av 100 sider og 0 av 27 sider som ble korrekt kategorisert. Årsakene til dette er at notesett 46 har over 100 sider og at stemme bare er angitt på hver 6. side. For notesett 47 er notene i liggende format. Ingen av disse oppsettene er støttet av vårt system. Det er flere slike tilfeller som dette, men som regel er det andre mindre årsaker som fører til feilkategoriseringer. Det kan for eksempel være mangel på alias i liste over godkjente stemmer og tett skrevet tekst. Det kan også være doble notesett, altså noter der flere instrument spiller samme stemme.

I tillegg kan en se i figur 13 resultatet for tolkning av tittel, komponist og arrangør hos de 68 notesettene vi har testet. Styreportalen ville også se på mulighetene for uthenting av denne informasjonen fra notesettene. Her er det lavere prosentmessig korrekt tolkede, henholdsvis 72%, 58% og 59%.

Antall Treff:	49	37	22
Antall Mulige:	68	64	37
Prosent treff:	72%	58%	59%

Figur 13 - Denne figuren viser antall korrekt tolket, fra venstre til høyre, tittel, komponist og arrangør.

6 DISKUSJON

I dette kapittelet vil vi se på resultatene fra kapittel 5, relatert til løsningen vi har valgt og hvordan programmet presterer i forhold til forventinger fra både oss som utviklere og Styreportalen som oppdragsgiver. Vi vil også diskutere begrensinger som ligger i løsningen slik den foreligger i dag og komme med noen forslag til forbedringer som vi mener vil kunne få programmet til å prestere enda bedre.

6.1 Utvikling av løsning

Som det fremgår i kapittel 5 hadde løsningen vår i de første testene et noe blandet resultat vurdert opp mot de evalueringsmetodene som vi bestemte oss for tidlig i prosjektet. Det har vært utfordinger underveis, slik som manglende tilganger hos Styreportalen, problemer med å få test-versjon av programmet til å fungere online og de generelle utfordringene ved å programmere i et nytt språk med annerledes syntax enn det vi hadde brukt før.

Løsningen vår ble at vi under utviklingsprosessen laget en type «mash-up» av Javascript og Python siden ingen av oss hadde vesentlig erfaring med utvikling i React og Node.js. I startfasen førte dette til at mye av algoritme-utiklingen gjort i Python som begge hadde erfaring med fra før. Denne måten å angripe oppgaven vår på viste seg å fungere bra og vi kom tidlig i gang med testing lokalt. Det gikk likevel med mye tid på å få opp en fungerende demoapp hvor både vi og andre kunne teste programmet online med egne notesett. Styreportalen ønsket denne demoappen laget i React, og vi brukte derfor mer tid enn ventet på å få dette på plass.

Mot slutten av utviklingsprosessen så vi muligheten for å overføre alle funksjoner og algoritmer til Node.js, slik at programmet enklest mulig kan integreres med resten av systemet til Styreportalen. På denne måten vil videre arbeid med optimalisering av programmet, samt vedlikeholdet gjøres mye enklere. Med mer tid til disposisjon hadde det vært ønskelig å overføre alle funksjoner til javascript.

6.2 Programmets prestasjon

Med alle justeringene og endringene vi har gjort underveis gjennom utviklingen og testing, har programmet nå en nøyaktighet på 77%. Årsakene til at programmet ikke har en 100% nøyaktighet vil vi se nærmere på nå.

En viktig årsak til at det oppstår feil når vi tester kategoriseringsverktøyet finner vi i hvordan outputtet fra Googles Vision API er bygget opp. Vårt program er avhengig av «bounding boxes» som Vision sender som output. Programmets algoritme bruker disse boksene til å filtrere mengdene ord den skal lete igjennom, for å redusere tid. Problemene med kategorisering oppstår da når programmet får med andre ting enn forventet i de boksene som den tolker. I slike tilfeller som dette tror vi en maskinlært modell kanskje hadde prestert bedre, men vi kan ikke bekrefte dette på noen måte nå da vi ikke har en modell å teste med.

Som vi så i kapittel 5.3 var det vanskelig å kategorisere noter som ikke var i riktig format. I tillegg oppstod det også problemer med kategorisering grunnet manglende aliaser, tett skrift og noter der flere instrument spiller samme stemme. Det er altså på disse områdene vi enklest kan sette inn tiltak for videre forbedring av programmet vi har utviklet. Problemene med format er det ikke mye vi får gjort noe med fra vår side da dette er en «feil» som kommer av eksterne faktorer ved å ikke ha noe standard for hvordan en note skal være. Som illustrert i figur 13 hadde vi to notesett som var vesentlig vanskeligere å katgorisere enn andre noter, nettopp på grunn av deres ikke-standard format. I det ene notesettet var notene rotert 90 grader, mens det andre notesettet kun hadde oppgitt stemme på hver 6. Side. I tillegg oversteg det ene notesettet grensen på 100 sider, som er maksimal lengde på en enkelt respons fra Vision API-et. Vi har ikke per nå tatt høyde for at sett på over 100 sider genererer flere responser. Disse faktorene førte derfor til at det ble spesielt mye feil på disse to notesettene.

46	Partita	18	100	Nei	Nei	Står kun stemmer på første av 6 sider per stemme. Over 100 sider, ikke støttet.
47	Ravenswood	0	27	Nei	Ja	Sidelengs noter, ikke støttet.

Figur 14 - Denne figuren viser resultat for kategorisering av 2 spesifikke notesett i test 2.

Det området hvor vi så mest forbedringspotensiale er økning av aliaser til de ulike stemmene. Dette er også det området som det kanskje er enklest å gjøre noe med. En av de vanligste feilene vi observerte under testing var mangel på alias i stemmeliste. Listen over godkjente stemmer som brukes av vårt program er bygd opp av de stemmer som vi annså som mest normale måter å skrive disse på. Det har vist seg at det veldig mange ulike måter å notere disse på. Det finnes ulike tradisjoner for dette, både fra land til land og fra forlag til forlag og at dette har endret seg over tid. Bare ved å legge til flere aliaser for godkjente stemmer vil ytelsen til programmet økes betraktelig.

Under testingen av Vision oppdaget vi raskt at denne «grensen» i hastighet ikke var noe vi kunne gjøre noe med. Uansett hvor små sidene vi matet inn i API-et var, brukte den like lang tid på å sende informasjon om siden tilbake. Det var altså ikke mulig for oss å få denne til å jobbe raskere. Men det som derimot er mulig, er å åpne for muligheten til å sende flere sider til kategorisering samtidig. I dokumentasjonen fra Google fremgår det at det fullt mulig å sende flere «requests» til Vision, som da åpner for multiprosessering av sidene i et notesett som skal kategoriseres. Denne funksjonen har vi dessverre ikke hatt tid til å sette oss inn i, men vi vet at potensialet for tidsbesparing her er svært høyt.

Styreportalen kom også med et ønske å få ut tittel, komponist og arrangør fra notesettene automatisk. Da dette ønsket ikke strengt talt var en del av oppgaven vår gruppe hadde blitt tildelt var det klart at ikke like mye fokus ville gå inn i å utvikle denne funksjonen.

I kapittel 5 observerte vi resultater for prosentandel korrekt tolket tittel, komponist og arrangør i test to. Her var de lavere prosenter sammenlignet med kategoriseringen av stemmer. Hovedårsaken til dette har vi funnet ut at ligger i hvordan vi tolker dataene fra Vision API-et. Ved stemmekategorisering kan vi bruke en predefinert liste over godkjente stemmer for å sjekke om ting blir korrekt kategorisert. Dette vil ikke fungere på hverken tittel, komponist eller arrangør da disse ikke er faste slik som stemmer, men heller varierer. Altså er det en jobb her å gjøre for å kunne korrekt identifisere disse hver gang i et notesett.

Vi tror her at maskinlæring ville hatt et bedre resultat enn den algoritmebaserte løsningen vårt program pr. nå tilbyr.

6.3 Begrensnigner

Som det fremgår i kapittel 5.3, så presterer programmet vårt ikke så bra som vi gjerne skulle ønsket i de tilfellene formatet på notene er forskjellig fra det som er mest vanlig. Dette gjelder særlig notesett i A3 format (tosidige), marsj-noter og noter som er av dårlig kvalitet/lav oppløsning, samt andre tilfeller der stemme, tittel og komponist/arrangør ikke står øverst tilvenstre, i midten øverst og øverst til høyre respektivt, målt i piksler.

For å forbedre programmet på disse områdene må det derfor lages nye, egne algoritmer som er spesifikt designet for å håndtere disse problemene. Denne mulgiheten ble diskutert med Sømoen, men han ønsket ikke dette da det ville skapt mye merarbeid for gruppen. Vanligvis er notesett tilgjengelig i A4-format, slik vi har designet vår algoritme for. Men som vi har sett betyr dette at det er enkelte typer notesett som programmet vårt ikke vil klare å kategorisere. Når det gjelder dårlige skannede noter vil ingen algoritme gjøre resultatet bedre da problemet ligger hos Googles Vision API som ikke korrekt klarer å hente ut dataene fra notesettet. Siden Styreportalen også ønsket at vi skulle bruke Vision i vår løsning, var det derfor begrenset hvilke forbedringer som var mulig å få til.

Et annet viktig moment er at løsningen vi har utviklet bare er basert på brassband-noter. Det vil si at listen over godkjente stemmer kun inneholder brassband-måter å skrive stemmer på. Dersom en ønsker å tilpasse programmet for bruk til janitsjarnoter må en da utvide listen som brukes av programmet til også å inkludere stemmer for en slik besetning.

7 KONKLUSJON OG VIDERE ARBEID

Utvikling av kategoriseringsverktøy for PDF-notesett.

I prosjektperioden har vi gjennom en iterativ prosess utviklet en løsning som kategoriserer PDF-notesett med en treffsikkerhet på ca. 77%. Styreportalen er fornøyd med resultatet og måten løsningen er laget på, da det er laget i Node.js og dermed er tilrettelagt for integrering mot deres allerede eksisterende systemer.

Løsningen er algoritmebasert og fungerer godt sammen med de data som den får fra Googles Vision API. Dette betyr samtidig at det fortsatt er rom for forbedringer til løsningen i form av maskinlærte modeller. Vi tror at dette ville kunne bidratt veldig positivt for den videre utviklingen av programmet. Opprinnelig tenkte vi at dette var noe som skulle være med i oppgaven, men det viste seg at tiden ikke strakk til for å teste ut dette. Ideelt sett hadde det beste vært om vi både hadde rukket å trene en modell til vårt formål, parallelt med at vi også arbeidet med vår algoritmebaserte løsning. Dersom en hadde kombinert begge disse metodene vil dette trolig gitt positive resultater, da en maskinlært modell vil kunne legge merke til mønstre og kjennetegn som mennesker ikke så lett oppdager (Géron, 2019). Ved å kombinere en maskins effektivitet i å kjenne igjen møstre sammen med det vi har programmert inn i vår løsning, mener vi at en kan oppnå svært høy nøyaktighetsgrad under kategoriserings-prosessen. I tillegg vil denne formen for løsning åpne for muligheten å eksludere bruken av Googles Vision API. Det er her systemet vårt bruker desidert mest tid og dermed også her den største optimeringsmuligheten ligger.

En annen forbedringsmulighet er å la brukere få tilgang til å oppdatere listen over godkjente stemmer som brukes i vår løsning og som består av godkjente, predefinerte stemmer og deres aliaser. Ved å gi brukere denne tilgangen vil Styreportalen kunne spare egne ressurser, da brukerene selv vil kunne være med på å forbedre programmets prestasjon. Men det er også utfordringer med en slik løsning. For eksempel vil en kunne tenke seg at noen vil legge inn stemmer som ikke faktisk er stemmer inn i listen. På en annen side ville dette kunne håndteres ved å etablere en rutine for gjennomgang og «rensking» av listen.

8 Referanser

Firebase, 2022. *Cloud Functions for Firebase.* [Internett]
Available at: https://firebase.google.com/docs/functions
[Funnet 18 Mai 2022].

Géron, A., 2019. *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow*. Sebastopol: O´Reilly Media, Inc..

Google Developers, 2022. *Firebase.* [Internett]
Available at: https://firebase.google.com/support/faq/

Google Developers, 2022. *Google Cloud.* [Internett]
Available at: https://cloud.google.com/vision

Google Developers, 2022. *Google Cloud AutoML.* [Internett]
Available at: https://cloud.google.com/automl

Google Developers, 2022. *Google Cloud Vision API.* [Internett]
Available at: https://cloud.google.com/vision/product-search/pricing

Google, 2021. *AnnotateImageResponse.* [Internett]
Available at:
https://cloud.google.com/vision/docs/reference/rest/v1/AnnotateImageResponse
[Funnet 19 Mai 2022].

Google, 2021. *AsynAnnotateFileRequest.* [Internett]
Available at:
https://cloud.google.com/vision/docs/reference/rest/v1/AsyncAnnotateFileRequest
[Funnet 19 Mai 2022].

Google, 2021. *BatchAnnotateFilesResponse.* [Internett]
Available at:
https://cloud.google.com/vision/docs/reference/rest/v1/BatchAnnotateFilesResponse
[Funnet 19 Mai 2022].

Google, 2022. *Authenticating to the Cloud Vision API.* [Internett]
Available at: https://cloud.google.com/vision/product-search/docs/auth
[Funnet 18 Mai 2022].

Google, 2022. *Batch file annotation offline.* [Internett]
Available at: https://cloud.google.com/vision/docs/file-batch#feature_detection_requests
[Funnet 18 Mai 2022].

Google, 2022. *Calling Cloud Functions.* [Internett]
Available at: https://cloud.google.com/functions/docs/calling
[Funnet 18 Mai 2022].

Google, 2022. *Detect text in files (PDF/TIFF).* [Internett]
Available at: https://cloud.google.com/vision/docs/pdf
[Funnet 18 Mai 2022].

Google, 2022. *OutputConfig | Cloud Vision API | Google Cloud.* [Internett]
Available at: https://cloud.google.com/vision/docs/reference/rest/v1/OutputConfig
[Funnet 18 Mai 2022].

Guerrero-Turrubiates, J. d. J., Gonzalez-Reyna, S. E., Ledesma-Orozco, S. E. & Avina-Cervantes, J. G., 2014. *Pitch Estimation For Musical Note Recognition Using Artificial Neural Networks.* Mexico, IEEE, pp. 53-58.

Jamshed, M., Maira, S., Rizwan, A. K. & And, M. U., 2020. Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR). *IEEE Access volume 8,* pp. 142642-142668.

Just Music UK, 2022. *Just Music Brass Band.* [Internett]
Available at:
https://www.justmusicuk.com/media/pdf/35000/JM35743%20Napoli.pdf?t=1651699725

Levenshtein, V. I., 1966. Binary codes capable of correctiong deletions, insertions, and reversals. *Soviet Physics-Doklady Vol.10, No. 8*, February, pp. 707-710.

Meta Platforms, Inc., 2022. *Reactjs.* [Internett]
Available at: https://reactjs.org

Nergaard, K., 2020. *Store Norske Leksikon Partitur.* [Internett]
Available at: https://snl.no/partitur
[Funnet 19 May 2022].

Norges Musikkkorps Forbund, 2022. *Musikkorps.no.* [Internett]
Available at: https://musikkorps.no/om-nmf-2/bli-medlem/meld-inn-korpset/

OpenJS Foundation, 2022. *Nodejs.* [Internett]
Available at: https://nodejs.org/en/about/

Salvendy, G. & Karwowski, W., 1997. *Handbook of Human Factors and Ergonomics,* New York: Wiley.

Schlipsing, M., Stallkamp, J., Salmen, J. & Igel, C., 2012. Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Neural Networks,* pp. 323-332.

Schwaber, K., 2022. SCRUM Development Process. *Buissness Object Desing and Implementation,* 18 May.pp. 117-134.

Sømoen, S., 2022. *Daglig leder Styreportalen AS* [Intervju] 2022.

Winkler, W. E., 1990. *String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage.*. [Internett]
Available at: https://files.eric.ed.gov/fulltext/ED325505.pdf

9 VEDLEGG

Prosjekthåndbok, visjonsdokument, kravdokument og systemdokumentasjon er vedlagt som separate filer

Resultat fra test med 10 notesett og test med 68 notesett

	funnet	total	kommentar	Tittel	Komponist		
Alt legger for din fot	28	28	mye doble stemmesett her, tok bar første stemme	bom	bom	treffprosent	75 %
Amazing grace	28	35	Bommet på partitur. Percussion ble perkusjon 1	treff	bom		
A night in Tunisia	29	30	Bommet på solo trombone, ikke lagt inn. Dobbletsidig	bom	bom	uten outliers	89 %
As the Deer	30	30	Flygehorn har samme stemme som 2. kornett.	treff	bom		
Barnard Castle	1	29	Dobbletsidig. Haukås logo oppe til venstre skapte problemer.	bom	bom		
Big Spender	23	24	Bommet på Mallets. Ikke lagt til i parts listen	treff	bom		
Bring back that Leroy Brown	48	51	Bommet på 2 slagverk stemmer, en Bb bass ble Eb bass, solo cornet ikke splittet.	treff	bom		
Caravan	53	67	Bommet på en del slagverk. Mangler side 2 og 3 med tuba.	treff	bom		
Christmas song	0	28	Bommet på alle, antakeligvis grunnet lang undertittel som strekker seg over hele siden	treff	bom		
Capriccio Brilliante	22	28	Bommet på andre side solo horn, sto ikke noen stemme der. Bommet på trommer og bass grunnet alias mangel				
sum	262	350					
sum uten helbom	261	293					

nummer	Tittel	sider korrekt	sider total	standard oppsett	Stemmer	kommentar	tittel	Identifisert komponist	arrangør	
1	Acht Klankstudies	18	25	Ja	Ja	Bommet på merkelig tuba notering, litt rot med horn og baryton, trolig løst med nytt alias	Ja	Ja	ingen	1
2	Alt Legger for Din fot	28	28	Ja	Ja	To stemmer skrevet på samme side for noen stemmer, gjenkjente kun en stemme	Nei	Nei	Ja	1
3	Amazing grace	29	33	Ja	Ja	Bommet på partitur, sto kun conductor på første side.	Ja	Nei	Nei	1
4	A night in Tunisia	30	30	Nei	Ja	dobbeltsidig	Nei	Nei	Ja	1
5	As the Deer	30	30	Ja	Ja	Dobbeltstemme for flygel og cornet 2, tok bare flygel da den hadde høyset score	Ja	Ingen	Ja	1
6	A psalm of praise	19	27	Nei	Ja	Dobbeltsidig. Lang undertittel gir tett tekst og større avsnitt	Nei	Nei	Nei	1
7	Activate	39	39	Ja	Ja	Standard oppsett	Ja	Ja	Ingen	1
8	Anthem	19	20	Ja	Ja	Bommet på siste percussion. Trolig grunnet mye tett tekst	Ja	Ja	Nei	1
9	Barnard Castle	29	29	Nei	Ja	Dobbeltsidig	Nei	Nei	Nei	1
10	Big Spender	23	24	Ja	Ja	Bom på Mallets, ikke lagt inn i parts	Ja	Ja	Nei	1
11	Bring back that Leroy Brown	47	51	Ja	Ja	Bom på Slagverk på slutten, problemer med solo kornett 1+2 og 3+4	Ja	Ja	Ja	1
12	Bruremarsj	12	13	Ja	Ja	Bom på flygelhorn. Tuba mangler forskjell på eb og bb, Bariton ikke splitteti 1 og 2	Ja	Ja	Ingen	1
13	Berceuse	27	29	Ja	Ja	Bom på Bb bass, Dårlig scan. Komponist og arrangør midt på siden.	Ja	Nei	Nei	1
14	BlackBird Special	19	29	Ja	Ja	Bom på en del side to hvor tittelen er skrevet med stemmen veldig tett. Klarer ikke hente ut stemmen alene	Ja	Ja	Ja	1
15	Boogie Down Partitur	21	21	Ja	Ja	Komponist i veldig liten skrift. Klarte ikke skille ut arr fra arrangør.	Ja	Nei	Ja	1
16	Brasilia	20	39	Ja	Nei	Sto ikke skrevet stemme etter side 1	Ja	Ja	Ingen	1
17	Caravan	59	67	Ja	Ja	Skjev skan gjorde at korte stemmer på side 2 og videre ikke ble regnet riktig	Ja	Ja	Ja	1
18	Christmas Song	27	28	Ja	Ja	Bommet på 1st horn, Uvvist hvorfor	Ja	Ja	Ingen	1
19	Capricio Brilliante	22	28	Ja	Ja	Bommet på solo side 2, ingen stemme skrevet. Bommet på Slagverkstemmer som ikke var lagt inn.	Ja	Ja	Nei	1
20	Chalk Farm no 2	27	27	Nei	Ja	Dobbeltsidig.	Nei	Nei	Ingen	1
21	Cappriccio	30	32	Ja	Ja	Bommet på solo stemme på starten.	Ja	Nei	Ingen	1
22	Circius	40	56	Ja	Ja	Bommet på side 2 på en del stemmer hvor Tittelen til stykket ble skrevet først	Ja	Nei	Ingen	1
23	Delig er Jorden	33	33	Ja	Ja	Noter ikke i partitur rekkefølge, Fremdeles OK	Ja	Ingen	Ja	1
24	Det er makt i de foldede hender	21	21	Ja	Ja	Bommet på komponist da tekst og musikk begge var nevnt, Cornet solo ble Cornet 1, Percussion ble Perkusjon 1	Ja	Nei	Ja	1
25	En stjerne skinner i natt	28	29	Ja	Ja	Bommet på Trommesett, alias "kit" ikke lagt inn.	Ja	Ja	Ja	1
26	The Essence of Time	90	90	Ja	Ja	Alt Ok	Ja	Ja	Ingen	1
27	Fra Borge	29	29	Ja	Ja	Trombone 1 og 2 på samme side	Ja	Ja	Ingen	1
28	How great Thou art	34	64	Nei	Ja	Dobbeltsidig, En del tomme sider	Nei	Nei	Ingen	1
29	Flesland Jubilee march	40	40	Ja	Ja	Perusjon 1 side 2 ble registret som perkusjon	Ja	Nei	Ingen	1
30	GaelForce	51	60	Nei	Ja	Dobbeltsidig, Mange tomme sider, tiltenkt tosidig utskrift, Mangler kit som alias for trommer.	Nei	Nei	Ingen	1
31	Gabriellas Song	25	30	Ja	Ja	Fikk opp clarinet solo et par ganger. Trolgi grunnet undertittle Cornet Solo. Rar feil	Ja	Ja	Ja	1
32	Himlen i min favn	21	39	Ja	Ja	Side 2 stemmene skrevet med tittel først, Bommet på alle disse	Ja	Ingen	Ja	1
33	Himmel på Jord	13	24	Nei	Ja	Blanding av Janitsjar og Brass Band stemmefordelign på alle noter.	Ja	Ja	Ja	1
34	I'd rather have Jesus	25	29	Ja	Ja	Undertittel med Cornet solo ga noen sider Clarinet Solo som stemme, Bedre alias løser dette	Ja	Nei	Ja	1
35	Irish tune from county Derry	22	22	Ja	Ja	Arrangør på venstre side, Flygelhorn sammen med repiano, ikke fanget opp-	Ja	Ja	Nei	1
36	Juleklokker	27	28	Nei	Ja	Tubular Bells oppfatter som tuba, Dobbeltsidig	Nei	Nei	Nei	1
37	Knut Liten og Sylvelin	50	56	Ja	Ja	Bommet på en del slagverksinstrumenter som ikke var lagt inn i parts listen, samt orgel.	Ja	Ja	Nei	1
38	Mitt hjerte alltid vanker	20	22	Ja	Ja	Bommet på sangstemme og Tubular Bells som ikke var lagt inn i parts	Ja	Ja	Nei	1
39	March - Dalarø	36	36	Ja	Ja	Alt Ok	Ja	Ja	Ingen	1
40	Midwest March	25	29	Ja	Ja	Mangler alias for E-flat bass og B-flat bass	Nei	Nei	Ingen	1
41	Misierlou	18	38	Ja	Nei	Står ikke stemme på side 2 av stemmene	Nei	Nei	Nei	1
42	On the castle Green	21	23	Nei	Ja	Dobbeltsidig, slagverkstemmer som ikke var lagt inn.	Nei	Nei	Ingen	1
43	O Magnus Mysterium	27	27	Ja	Ja	Alt Ok	Ja	Ja	Ja	1
44	Over the rainbow	26	27	Ja	Ja	Bommet på første solo horn nytt alias bør løse	Ja	Ja	Ja	1
45	Praise	11	25	Ja	Ja	Bommet på en del stemmer, Trolig tett tekst som utgjør problemet	Nei	Ja	Ingen	1
46	Partita	18	100	Nei	Nei	Står kun stemmer på første av 6 sider per stemme. Over 100 sider, ikke støttet.	Ja	Ja	Ingen	1
47	Ravenswood	0	27	Nei	Ja	Sidelengs noter, ikke støttet.	Nei	Nei	Ingen	1
48	Slaidburn	13	18	Nei	Ja	Marsjnoter	Ja	Ja	Ingen	1
49	Spain	56	68	Ja	Ja	Bommet på partitur	Ja	Ja	Ja	1
50	Shine as the light	17	45	Ja	Nei	Ikke skrevet stemme etter side 1 åer stemme, Litt bom på slagverk	Ja	Nei	Ingen	1
51	Silent Night	24	25	Ja	Ja	Mangler stemme på side 2 Euph.	Ja	Ingen	Ja	1
52	Diversions	19	37	Nei	Nei	Tosidig	Nei	Nei	Ingen	1
53	Shenendoah	10	15	Ja	Ja	Bommet på partitur	Ja	Ja	Ja	1
54	Solo Secondo	26	47	Nei	Ja	Bommet på partitur, en del stemmer hvor det står skrevet flere stemmer på samme side. Stemmer i ulike nøkler	Ja	Ja	Ingen	1
55	Prelude to prayer (spirit of life)	14	22	Ja	Ja	Partitur sidelengs. Bommet på 1. og 2. horn som kom på samme side	Ja	Ja	Ingen	1
56	Storbystev	17	20	Ja	Ja	Bb bass og Eb bass byttet plass, uvvist hvorfor. Mangler alias kit til drums.	Ja	Ja	Ingen	1
57	Sunrise over blue ridge	28	29	Ja	Ja	Bommet på Perkusjon 1 side 2	Ja	Ja	Ingen	1
58	March - The Red Shield	36	43	Ja	Ja	Bommet på Perkusjon etter første side	Ja	Ja	Ingen	1
59	Tocatta from cats tales	27	28	Nei	Ja	Tosidig, bommet på siste perkusjon stemme	Nei	Nei	Ingen	1
60	Two Christmas Scenes	29	29	Ja	Ja	Alt OK	Ja	Ja	Ingen	1
61	Trittico	33	58	Nei	Nei	Tosidig, Bommet på andre side av stemmene	Nei	Nei	Ingen	1
62	The Witch of the Westmerlands	21	28	Ja	Ja	Bommet på en del aliaser som manglet, Fikses ved å legge til i parts listen	Nei	Nei	Nei	1
63	Call of the righteous	25	26	Ja	Ja	Bommet på Ess kornet i starten, Uvisst hvorfor	Nei	Nei	Nei	1
64	Til Ungdommen Din tanke er fri	7	16	Nei	Ja	Marsjnoter, mye doble stemmer på samme side	Ja	Ja	Ja	1
65	Torn-Eriks song	19	39	Ja	Nei	Står ikke stemme på side 2 av stemmene, Bommet på en perk stemme og solo i starten	Ja	Nei	Ja	1
66	Valdresmarsj	18	19	Ja	Ja	Mangler skarptromme i stemmelisten.	Ja	Ja	Ja	1
67	Windows of the World	27	86	Nei	Nei	Tosidig	Nei	Ja	Nei	1
68	World Dances	61	62	Ja	Ja	Bommet på en perkusjonsstemme, ble perk 1 i stedet for perkusjon	Ja	Ja	Ingen	1
	Sum	1853	###		77 %					
		1432	###		85 %					

Antall Treff:	49	37	22
Antall Mulige:	68	64	37
Prosent treff:	72 %	58 %	59 %