UNIVERSITETET I BERGEN

Masteroppgave

Informasjonsgjenfinning og visualisering av relasjoner i norsk popmusikk

---

Anders Langseth

1. juni 2015

Abstrakt

Utforskning i og tilgang til innhold i norsk popmusikk er et stort interesseområde med flere aktive aktører. Akademiske, kommersielle og private aktører er interessert i å forstå norsk popmusikk, oppdage interessante mønstre og utforske relasjoner mellom sanger. For å realisere dette kreves det ofte menneskelig analyse av flere tusener av sanger, noe som er en tidskrevende oppgave. Ved å bruke gjenfinningsmetoder for å finne relasjoner og å visualisere de i et nytt grensesnitt kan det hjelpe brukeren i å analysere og resonnere om de data presentert. Dette gjør komplekse data mer tilgjengelig, forståelig og brukervennlig for de som er interessert i norsk popmusikk.

Avhandlingen beskriver utviklingen av en prototype av et system som støtter gjenfinning og visualisering av relasjoner i norsk popmusikk. For å oppdage relasjoner bruker systemet metodene latent semantic indexing(LSI), termfrekvens-invers dokumentfrekvens(TF-IDF) og en egendesignet algoritme til å analysere sangtekster og akustiske egenskaper. For å presentere relasjoner visualiseres data til meningsfulle grensesnitt i en web-applikasjon. Grensesnittene kan brukes til å oppdage og utforske norsk popmusikk, eller bare for å finne nye sanger å lytte til. Web-applikasjonen evalueres ved usability-testing og resulterte i test-deltakere som var tilfredse med applikasjonen og enkelte problemer ble oppdaget. Avhandlingen evaluerer også bruken av gjenfinningsmetoder for sangtekster i systemet, hvor TF-IDF presterer bedre enn LSI.

Forord

Jeg vil rette en stor takk til professor Weiqin Chen, for å rådgi dette prosjektet og hennes mange nyttige råd gjennom prosjektet.

Jeg vil også takke Varun Jewalikar hos Musixmatch for å gi systemet tilgang til deres API via et eget endpoint, Barry Fitzgerald for Angular directivet highcharts-ng som forenklet samspillet mellom Angular og Highcharts, Jonathan Hedley for den svært nyttige jsoup HTML-parseren og alle testdeltakere som har bidratt til testing av prototypen.

Innholdsfortegnelse

Liste over Figurer

Liste over Tabeller

Liste over Kodesnutter

Nomenklatur

API – Applikasjonsprogrammeringsgrensesnitt

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

IDF - Invers dokumentfrekvens

LSI - Latent Semantic Indexing

SVD - Singulær-verdi dekomposisjon

TF - Termfrekvens

TF-IDF - Term Frekvens-Invers Dokument Frekvens

SUS – System Usability Scale

1 Introduksjon

Den raske utviklingen av Internett, bærbare enheter, multimedia komprimering og lagringsteknologi har i stor grad gjort det mulig å ta med seg musikk hvor som helst. I Norge i dag kan man høre musikk så å si overalt: det blir hørt i butikker, restauranter, konsertlokaler, treningsstudioer, på jobb og mange flere steder. Musikk er derfor en sentral del i flere nordmenn sine liv.
Å kunne utforske og få en forståelse av musikken nordmenn hører på(norsk popmusikk) er et stort interesseområde med flere aktive aktører. Enkeltpersoner, forskere og plateselskap har alle ulike grunner til å ville få en oversikt over norsk popmusikk og kunne sammenligne sanger i kolleksjonen. For enkeltpersoner kan informasjonen benyttes til å få en oversikt over sanger og finne ny musikk. Forskere kan bruke informasjonen til å svare på interessante spørsmål om kulturelle forskjeller eller tidsepoker i norsk historie. Plateselskap og kommersielle-aktører kan oppdage interessante trender i musikken slik at de vet hva slags type musikk som er populær, og avgjøre hvilke sanger de bør støtte og promotere.

I dagens verden er data om norsk popmusikk tilgjengelig på VG-Lista Topp 20 sitt nettsted[1]. Nettstedet presenterer norsk popmusikk sin historie fra 1958 og frem til i dag og inneholder interessant informasjon om ca. 3, 000 lister og 5, 000 unike sanger. På nettstedet kan man enten utforske sanger ved å trykke seg gjennom alle listene eller ved å søke i søkemotoren. Å bruke en søkemotor som nesten den eneste måten å oppdage ny musikk på får ofte brukere til å føle seg frustrert. Dette er fordi hvis man ønsker å sammenligne eller oppdage en sang er det ikke alle som er klar over tittelen, artist eller album. Dette støtter ikke oppdagelse og utforskning av sanger slik som flere brukere ønsker, og er et kjent problem i store digitale musikkolleksjoner.

For å kunne utvikle et system som kan bli brukt til å tilfredsstille informasjonsbehovene til de forskjellige målgruppene nevnt ovenfor må systemet sitt datasett være basert på norsk popmusikk og systemet må håndtere to utfordringer. Disse er å identifisere mål for likhet mellom sanger på en måte som brukere er enige i og presentere data på en intuitiv måte som letter analysen av det komplekse datasettet.

Å identifisere likhet mellom sanger er en enkel oppgave for mennesker.
Når vi snakker om musikk og prøver å forklare egenskapene til andre, beskriver vi ofte hvor lik en

[1] http://lista.vg.no/

sang er en annen. En årsak til dette er at det er mye lettere å forestille seg hvordan en sang kan høres ut om vi kan knytte det til en sang vi allerede kjenner til(Hilliges, Holzer, Klüber, & Butz, 2006, s. 1). Å lære et system å sammenligne sanger på en tilnærmet lik måte er en vanskelig oppgave. I dag brukes det flere forskjellige tilnærminger for å identifisere likhet mellom sanger. En av de mest brukte tilnærmingene er å organisere musikkolleksjoner etter forhåndsdefinerte sjangre som klassisk, metall, rap, rock, men dette hjelper ikke veldig mye hvis man ønsker å oppdage sanger i en musikkolleksjon på 5, 000 sanger. Dessuten er det en tidkrevende prosess for musikk-eksperter å skulle kategorisere så mange sanger.

I denne avhandlingen brukes det to tilnærminger som ikke er avhengig av ytterligere informasjon, slik som forhåndsdefinerte kategorier. Istedenfor kan brukere oppdage sanger ut ifra to forskjellige mål for likhet som baserer seg på sangenes innhold, slik som akustiske egenskaper og sangtekst.

For å presentere likhet mellom sanger og andre interessante mønstre i norsk popmusikk, presenteres data i forskjellige visuelle komponenter som gjør det enkelt å oppdage og utforske data. Datavisualisering av komplekse data lar brukeren få innsikt og gir de muligheten til å trekke konklusjoner, ved å direkte samhandle med data.

1.1 Forskningsspørsmål

Forskningsspørsmålet for denne avhandlingen er:

"Hvordan støtte gjenfinning og visualisering av relasjoner i norsk popmusikk?"

For å besvare forskningsspørsmålet har jeg utviklet og testet et system for å støtte gjenfinning og visualisering av relasjoner i norsk popmusikk. Systemet består av en tung back-end som står for datainnsamling og gjenfinningsmetoder basert på sangers innhold, og en Web front-end for å visualisere relasjonene i datasettet. Datasettet som er brukt for å representere norsk popmusikk er sanger på VG-Lista Topp 20, fra år 1960 til og med 2014.

1.2 Mål

Hovedmålet bak prosjektet var å designe og implementere et «proof of concept» system for å støtte gjenfinning og visualisering av relasjoner i norsk popmusikk. Dette vil si å oppdage mønstre og relasjoner i selve musikkolleksjonen, men også blant artister, lister og sanger. Mønstre i kolleksjonen presenteres over år, slik at man får et bilde over endringen norsk popmusikk har gjennomgått. For å oppdage relasjoner mellom artister, lister og sanger er det brukt gjenfinningsmetoder for å finne like sanger basert på tekstlig innhold og en egendesignet algoritme for akustisk likhet.

Den visuelle delen av systemet er implementert som en web-applikasjon og er designet for å være brukervennlig og lett tilgjengelig. Lett tilgjengelig vil si at applikasjonen skal lastes raskt og være tilgjengelig på Internett, slik at det ikke er nødvendig å laste ned tilleggsverktøy for å benytte seg av datavisualiseringene. Brukervennlighet er også viktig for web-applikasjonen, slik at brukere enkelt kan forstå datasettet via interaktive komponenter som skal være responsive til brukeren sine handlinger.

Det store målet med systemet er at data som presenteres vil kunne bli brukt til å svare på spørsmål om norsk popmusikk og muligens det norske folk fra 1960-2014. Det er viktig å påpeke at avhandlingen ikke har som mål å finne mønstre i norsk popmusikk og presentere de, men heller å tilrettelegge for at andre kan bruke systemet til videre forskning innen forskjellige fagområder eller bare for å oppdage ny musikk. De data systemet presenterer kan derfor være av interesse for musikkinteresserte personer, historikere, journalister, produsenter, låtskrivere, med mer.

1.3 Organisering av avhandlingen

Kapittel 2 gir en oversikt over fagområdene informasjonsgjenfinning og datavisualisering. Innenfor fagområdet informasjonsgjenfinning introduseres det forskning relatert til to gjenfinningsmetoder, latent semantic indexing(LSI) og termfrekvens-invers dokumentfrekvens(TF-IDF). Videre presenteres fagområdet datavisualisering, hvilke fordeler det innebærer å visualisere komplekse datasett og forskning gjort innen visualisering av musikkolleksjoner.
Kapittel 3 presenterer hvilken forskningsmetodikk som er benyttet og hvordan den er blitt brukt som et rammeverk for å sikre god teknologi basert forskning.

Kapittel 4 forklarer hvilken utviklingsmetodikk som er brukt og hvilke iterasjoner systemet er utviklet gjennom, fra planlegging av systemet og til visualisering av innsamlet data.
Kapittel 5 evaluerer systemet gjennom to typer evaluering. Evaluering av gjenfinningsmetodene LSI og TF-IDF, for å finne den metoden som presterte best i systemet og evaluering av brukervennligheten til systemet sin web-applikasjon, ved usability-testing.
Til slutt, i kapittel 6, trekkes konklusjoner, avhandlingen reflekteres og det gis et innblikk i fremtid arbeid.

2 Teori

Dette kapittelet tar for seg noe av bakgrunnen og forskningen gjort i de mest sentrale fagområdene for prosjektet, datavisualisering og informasjonsgjenfinning. System- og webutvikling er også veldig sentralt for prosjektet, men har ikke blitt prioritert i denne delen, ettersom de ikke særpreger prosjektet på lik linje som datavisualisering og informasjonsgjenfinning.

Forskning av tidligere arbeid har i oppstartsfasen vært viktig for å idemyldre, kartlegge og å få en generell oversikt over hva som er gjort innenfor de forskjellige fagområdene. Noen teorier presentert i dette kapittelet er spesifikke og inneholder avanserte formler, mens andre er mer generelle og beskriver ideer.

2.1 Informasjonsgjenfinning

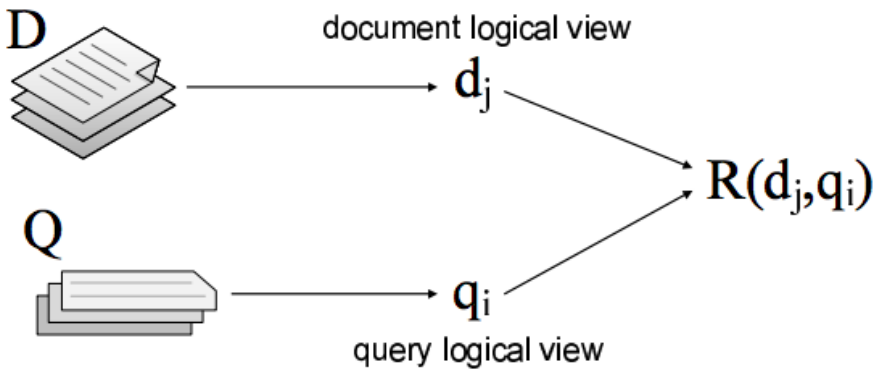
Informasjonsgjenfinning(Information Retrieval) er mest kjent som aktiviteten av å innhente informasjon som er relevant til en bruker sitt informasjonsbehov. IR håndterer også representasjonen, lagring, organisering av, og tilgang til informasjonselementer som nettsider, strukturert og semi-strukturerte data og multimedia objekter. Omfanget til IR har vokst langt utover sine tidligere mål som indeksering av tekst og søk etter nyttige dokument i en kolleksjon. I dag inneholder fagområdet forskning innen modellering, søkemotorer, tekstklassifisering, systemarkitektur, brukergrensesnitt, datavisualisering, filtrering og språk (Baeza-Yates & Ribeiro-Neto, 2011, s. 1).

I forskning baserer IR seg ofte på å finne relevante dokumenter til en gitt brukergenerert spørring. I

det implementerte informasjonssystemet skal det finnes relevante dokumenter ut ifra et annen dokument i fra samlingen. Dette fører ikke til en annen tilnærming av IR, ettersom alle spørringer blir håndtert som et dokument i IR. Derfor er det greit å tolke ordet "spørring" som vil bli brukt videre i denne avhandlingen som et annet dokument(sang) i fra samlingen.

For å forutsi hvilke dokumenter som er like, må informasjonsgjenfinningssystemet forholde seg til ett sentralt problem – å forutsi hvilke dokumenter brukerne finner relevante og hvilke de finner irrelevante på bakgrunn av en spørring. Dette bærer en viss uklarhet eller usikkerhet, ettersom brukere kan ha forskjellige meninger om hvilke dokumenter som er relevante. For å løse dette bruker IR-systemer en prediktiv algoritme som forhåpentligvis tilnærmer meningene til de fleste brukere. Denne prediktive algoritmen kalles en rangeringsfunksjon og blir brukt til å etablere en sortering av dokumenter returnert. Rangeringsfunksjonen tildeler en poengsum til dokumenter med hensyn til spørringen (Baeza-Yates & Ribeiro-Neto, 2011, s. 58). Denne prosessen består av to deler:

- Ideen av et logisk rammeverk for å representere dokumenter og spørringer, kalt modell.
- Definisjonen av en rangeringsfunksjon, som beregner rangering for hvert dokument med hensyn til spørringen.



Figur 2.1 Forholdet mellom modell og rangeringsfunksjon fra Baeza-Yates & Ribeiro-Neto(2011, s. 59)

Figur 2.1 viser dokumentet D, spørringen Q og sammenhengen mellom:

- Modellen(logical view) til dokument d_j og spørring q_i
- Rangeringsfunksjonen $R(d_j, q_i)$.

Gitt en modell for dokument og spørring, slik som d_j og q_i, vil rangeringsfunksjonen $R(d_j, q_i)$ tildele en rangering(nummer), til dokument d_j med hensyn til spørring q_i.

Som illustrert i Figur 2.1, vil modellen direkte påvirke beregningen av dokumentet D sin rangering i rangeringsfunksjonen $R(d_j, q_i)$. Det er derfor viktig å velge modell og rangeringsfunksjon som gir gode resultater i en gitt kolleksjon.
De tre klassiske modellene i IR er boolsk-, vektor- og probabilistiskmodell. I den boolskemodellen er dokument og spørring representert som mengder av ord, vektormodellen representerer de som vektorer i et t-dimensjonalt rom og i den probabilistiske-modellen er de representert som sannsynligheter. De modellene som presenteres i dybden er den klassiske vektormodellen og en variant av vektormodellen, kalt semantic indexing-modellen(Baeza-Yates & Ribeiro-Neto, 2011, s. 59). Disse to modellene er valgt på bakgrunn av deres to forskjellige tilnærminger av måten å gjenfinne dokumenter på. TF-IDF sin tilnærming er leksikalsk og vil si at den sammenligner dokumenter på ord og antar at de betyr det samme i de forskjellige dokumentene. LSI sin tilnærming er semantisk og muliggjør sammenligning av dokumenter på basis av et konseptuelt tema eller betydningen av et dokument.

2.1.1 **Vektormodellen og TF-IDF**

Vektormodellen ble antatt først introdusert i SMART (System for the Mechanical Analysis and Retrieval of Text) Information Retrieval System som er et informasjonsgjenfinningssystem utviklet ved Cornell University i 1960-årene (Manning & Raghavan, 2009, s. 133). Modellen er et svar på ulemper i den boolske modellen. Den boolske modellen forutsier at hvert dokument enten er relevant eller ikke relevant. Det vil si at det ikke er mulig med en delvis likhet mellom dokument og spørring. På grunn av dette binære beslutningskriteriet, som ikke har noen gradering av likhet, fører det ofte til at enten for mange eller for få dokumenter blir foreslått som like. Vektormodellen løser dette ved å foreslå en modell hvor delvis likhet er mulig.

Dette gjøres ved å tildele ord i dokumenter og spørring en ikke-binær vekt, som tilslutt brukes for å beregne graden av likhet mellom hvert dokument og spørringen.
Ved å så sortere de returnerte dokumentene i synkende grad av likhet, tar vektormodellen hensyn til dokumenter som samsvarer delvis med termene i spørringen. Resultatet er at rangerte dokumenter gir mer nøyaktige svar på spørringer enn ved bruk av den boolskemodellen (Baeza-Yates & Ribeiro-Neto, 2011, s. 77).

For å kunne benytte en rangeringsfunksjon til å finne graden av likhet mellom dokument og

spørring må termene vektes med en ikke-binær verdi. I den klassiske vektormodellen vektes de oftest med termfrekvens(TF), invers dokumentfrekvens(IDF) eller produktet av termfrekvens og invers dokumentfrekvens, også kalt TF-IDF. Disse tre metodene kan representeres på flere matematiske måter, men tanken og nytten bak de vil fremdeles være den samme. Videre introduseres forskning og mulige implementasjoner av de.

Termfrekvens

Vekting av termer med termfrekvens er basert på Luhn (1957) sin antakelse om at verdien, eller vekten, av term t_i som forekommer i et dokument d_j er ganske enkelt proporsjonal til termfrekvensen $f_{i,j}$. Det vil si, jo oftere en term t_i oppstår i teksten til dokument d_j, jo høyere er dens term frekvens vekt $tf_{i,j}$ (Baeza-Yates & Ribeiro-Neto, 2011, s. 68).
Antakelsen er basert på observasjonen av at termer som oppstår ofte er viktige for å beskrive emnene i et dokument. Dette fører til følgende formel for termfrekvens:

$$tf_{i,j} = f_{i,j}$$

Et eksempel av TF-vekting for en mengde dokumenter er illustrert i Tabell 2.1.

	D1	D2	D3	D4
Angeles	0	0	1	0
Los	0	0	1	0
New	1	1	0	1
Post	0	1	0	2
Times	1	0	1	0
York	1	1	0	0

Tabell 2.1: Ren TF-matrise

Ved å vekte TF på denne måten vil lengre dokumenter ha en fordel i gjenfinning over korte dokumenter. Dette er fordi lengre dokumenter vanligvis bruker samme term gjentatte ganger, som fører til at termfrekvensen sin vekt for en term vil være større for lengre dokumenter enn korte dokumenter(Baeza-Yates & Ribeiro-Neto, 2011, s. 75). De lengre dokumentene inneholder også ofte flere unike termer, noe som vil øke sannsynligheten for at et lengre dokument er relevant istedenfor et kort dokument. For å løse dette problemet innføres dokumentlengde

normalisering(Document length normalization) som er en måte å straffe termvekter for et dokument i samsvar med sin lengde. Det finnes flere måter å representere dette på. En måte er å dele TF-vekten for en term med antall forekomster av termer i dokumentet (Baeza-Yates & Ribeiro-Neto, 2011, s. 76). Si hvis et dokument har to ord, et som forekommer to ganger, og et annet som forekommer tre ganger, vil det første ordet bli normalisert til 2/5 (0,4) og det andre til 3/5 (0,6). Eksempelet i Tabell 2.2 inneholder dokumenter med like antall ord, så normalisering av dokumentlengden er ikke i stor grad nødvendig, men det illustrerer resultatene av utregningene.

	D1	D2	D3	D4
Angeles	0	0	0.33	0
Los	0	0	0.33	0
New	0.33	0.33	0	0.33
Post	0	0.33	0	0.66
Times	0.33	0	0.33	0
York	0.33	0.33	0	0

Tabell 2.2: Dokumentlengde normalisert TF-matrise

Invers dokumentfrekvens

Termfrekvensen lider av et kritisk problem: alle termer anses som like viktige når det gjelder å vurdere relevansen til en spørring. Dette er ikke optimalt, ettersom alle termer i et dokument vil ikke være like nyttige for å beskrive innholdet. Baeza-Yates & Ribeiro-Neto (2011, s. 66) gir følgende eksempel som beskriver problemet: "Anta en samling med hundre tusen dokumenter. Et ord som finnes i alle dokumentene er ikke nyttig fordi det ikke forteller oss noe om hvilke dokumenter brukeren kan være interessert i. På en annen side, vil de ordene som vises i få dokumenter i samlingen være nyttige, ettersom de begrenser mengden av dokumenter som brukeren kan være interessert i". Derfor bør det regnes med at forskjellige termer har varierende betydning/viktighet når de brukes til å beskrive dokumentet sitt innhold.
Jones (1972) introduserte en metode for å kalkulere termen sin viktighet, kalt invers-dokumentfrekvens. Hun beskriver IDF som at termen sin viktighet kan bli kvantifisert som en invers funksjon av antall dokumenter hvor den oppstår (Baeza-Yates & Ribeiro-Neto, 2011, s. 71). Dette resulterte i formelen for IDF: $idf_i = \log \frac{N}{|\{d \in D : t_i \in d\}|}$

Hvor N er antall dokumenter i samlingen og $|\{d \in D : t_i \in d\}|$ er antall dokumenter d i kolleksjonen D hvor termen t_i oppstår. Logaritmen av dette fører til at IDF-vekten til en sjelden

term er høy, mens IDF-vekten til en hyppig term er sannsynlig å være lav.
Et eksempel av IDF-vekting er illustrert i Tabell 2.3.

	D1	D2	D3	D4
Angeles	0	0	2	0
Los	0	0	2	0
New	0.415	0.415	0	0.415
Post	0	1	0	1
Times	1	0	1	0
York	1	1	0	0

Tabell 2.3: IDF-matrise

Hvilken base av logaritmen man bruker utgjør ingen forskjell for vekting. I Tabell 2.3 er det blitt brukt log2.

Termfrekvens – Invers dokumentfrekvens

Ikke lenge etter forslaget om IDF, foreslo Salton og Yang (1973) at man kunne kombinere termfrekvens og invers dokumentfrekvens, noe som i ettertid har vist seg å forbedre vekting av termer. Funksjonen de foreslo er blitt kalt TF-IDF og er blitt den mest populære metoden å vekte termer på (Baeza-Yates & Ribeiro-Neto, 2011, s. 72).
Den enkleste formelen for TF-IDF er:

$$tfidf_{i,j} = tf_{i,j} \times idf_i$$

Hvor $tf_{i,j}$ representerer termfrekvensen for termen i i dokument j og idf_i representerer den inverse dokumentfrekvensen til termen i. Produktet av termfrekvens og invers dokumentfrekvens for en term resulterer i TF-IDF. En mulig utregning av TF-IDF er presentert i Tabell 2.4.

	D1	D2	D3	D4
Angeles	0	0	0.66	0
Los	0	0	0.66	0
New	0.14	0.14	0	0.14
Post	0	0.33	0	0.66

Times	0.33	0	0.33	0
York	0.33	0.33	0	0

Tabell 2.4: TF-IDF-matrise

I Tabell 2.4 er TF-IDF utregnet og man ser at TF-IDF vekten til en term er høy hvis termen oppstår flere ganger i få dokumenter og lavere hvis den oppstår færre ganger i et dokument, eller mange ganger i flere dokument. Lavest TF-IDF vekt er når termen oppstår i ingen av dokumentene, altså 0.

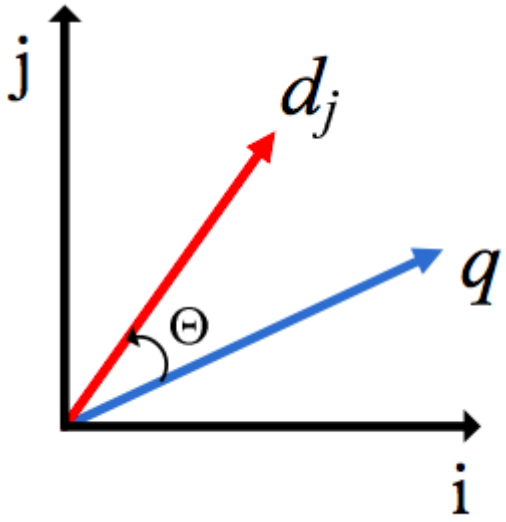
Cosinus-likhet

Videre i vektormodellen brukes en av de tre vektene introdusert for å representere dokument d og spørring q som vektorer i et t-dimensjonalt vektorrom, hvor t representerer antall unike termer i kolleksjonen. For dokumentet j vil hver dimensjon inneholde vekten $w_{i,j}$ for den gitte termen i i dokumentet j. Vektoren til dokument d_j vil betegnes som $\overrightarrow{d_j}$.
Den matematiske representasjonen for vektorene til dokument $d_j$og spørring q er:

$$\overrightarrow{dj} = (w_{1,j}, w_{2,j}, w_{3,j}, \dots, w_{t,j})$$
$$\overrightarrow{q} = (w_{1,q}, w_{2,q}, w_{3,q}, \dots, w_{t,q})$$

Hvor vektene $w_{i,j}$ & $w_{i,q} \geq 0$.

Å representere dokument d_j og spørring q som vektorer gjør det mulig å representere de i et vektorrom, illustrert i Figur 2.2.



Figur 2.2: Vektorrom representasjon av dokument d_j og spørring q

I vektorrommet er det mulig å beregne en likhets koeffisient mellom vektorene, som gjenspeiler graden av likhet i deres vekter(dimensjoner). Dette gjøres ved å regne ut cosinus mellom de to vektorene. Cosinusverdien forteller noe om forholdet mellom de to vektorene. Verdien blir kalkulert ut ifra antall grader vinkel det er mellom de to vektorene og kan være fra -1 og til 1. Hvor -1 vil si at vektorene er motstående og 1 vil si at vektorene har en vinkel på 0°. Følgende er en implementasjon for å regne ut cosinus-likheten mellom to vektorer:

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|}$$

Hvor $\vec{d_j} \cdot \vec{q}$ er det euklidske indreprodukt av de to vektorene, altså produktet av vektene $w_{i,j}$ og $w_{i,q}$ for alle vekter i de to vektorene, og $|\vec{d_j}| \times |\vec{q}|$ er produktet av den euklidske norm til d_j og q. Den euklidske norm er kvadratroten av summen av vektene opphøyd i 2e for den gitte vektoren. Formelen er gitt ved:

$$|\vec{x}| = \sqrt{x_{1^2} + \cdots + x_{t^2}}$$

Hvor x er vektor representasjonen av dokument d_j eller spørring q og t er antall unike termer i kolleksjonen.

Cosinus-likhet funksjonen returnerer cosinusverdien, som er et desimaltall, mellom vinkelen av de

to vektorene, dokument $d_j$og spørring q. Ettersom det tidligere ble definert at vektenes verdi ikke er mindre enn 0, $w_{i,j}$ & $w_{i,q} \geq 0$, så vil aldri vektorene kunne være motstående, slik at cosinus verdien er under 0. Resultatet av cosinus-likhets funksjonen, $sim(d_j, q)$, returnerer derfor en cosinus verdi mellom 0 og 1. Eksempelvis vil cosinusverdien mellom de to vektorene for dokument $d_j$og spørring q med 0° resultere i 1, som vil si at de er helt like. Hvis gradene mellom vektorene er over 90° vil metoden returnere 0, som tilsier at de er helt ulike. Hver cosinus verdi mellom 0 og 1 forteller noe om hvor like de er og muliggjør en verdi for delvis likhet mellom vektorer.

Ifølge Baeza-Yates & Ribeiro-Neto (2011, s. 79) er de store fordelene ved vektormodellen at:

- Vekting av termer forbedrer kvaliteten av likhet
- Delvis likhet tillater resultater som er omtrentlig like spørringene.
- Cosinus-likhets funksjonen sorterer dokumenter i deres grad av likhet i henhold til spørringen.

Til tross for sin enkelhet, er vektormodellen en motstandsdyktig rangerings-strategi for generelle dokument kolleksjoner. Det gir rangerte resultater som er vanskelig å forbedre uten query-expansion eller relevance feedback. Et stort utvalg av alternative rangeringsmetoder har blitt sammenlignet med vektormodellen, men konsensus betrakter modellen å være en enkel, god og rask rangeringsmetode. Av disse grunner, fortsetter vektormodellen å være en populær modell, som stadig anvendes som ”ground truth” ved evaluering av alternative rangeringsfunksjoner og nylig foreslåtte modeller(Baeza-Yates & Ribeiro-Neto, 2011, s. 79).

2.1.2 **Latent Semantic Indexing-modellen**

Latent semantic indexing-modellen bygger på vektormodellen ved at den modellerer vekter ved en reduksjonsteknikk kalt singulærverdi-dekomposisjon(SVD). Modellen ble oppfunnet av Scott Deerwester, Susan Dumais, George Furnas, Richard Harshman, Thomas Landauer, Karen Lochbaum og Lynn Streeter i 1990 og er et svar på ulemper i den klassiske vektormodellen.

Den klassiske vektormodellen baserer seg på en leksikalsk tilnærming til å finne relasjoner mellom dokumenter. Dette vil si at de matcher dokumenter på ord og antar at de betyr det samme i de forskjellige dokumentene. Ettersom det er mange måter å uttrykke et gitt konsept på(synonym) kan dette føre til at et dokument som uttrykker det samme med andre ord ikke vil være relevant. I tillegg

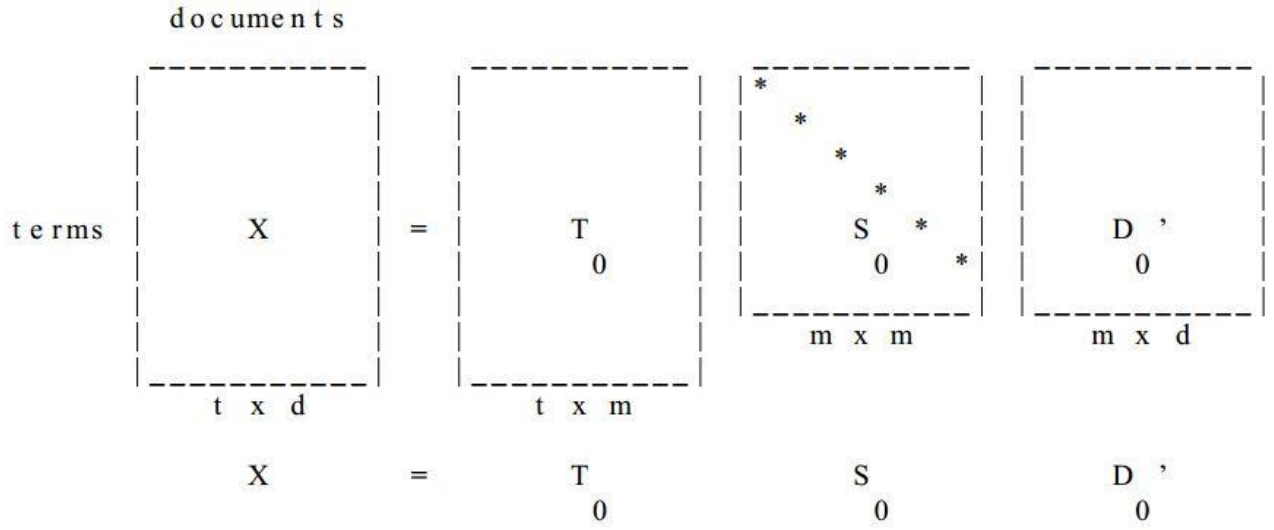
kan flere ord ha flere betydninger(polysem), noe som kan resultere i at irrelevante dokumenter presenteres som relevante (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990, s. 1). LSI er designet for å løse akkurat disse problemene ved å muliggjøre sammenligning av dokumenter på basis av et konseptuelt tema eller betydningen av et dokument. Tilnærmingen LSI bruker er at den antar at det er en underliggende semantisk struktur i dataene som er delvis skjult av tilfeldigheten av ordvalg. For å finne relaterte dokumenter baserer LSI seg på statistiske metoder for å estimere likheten, ved å kvitte seg med "støy" (Deerwester et al., 1990, s. 1).

For å kvitte seg med "støyen" benytter latent semantic indexing seg av singulærverdi-dekomposisjon. SVD tar en matrise av vektede ord-dokument assosiasjoner som TF, IDF eller TF-IDF og konstruerer et semantisk "rom" hvor termer og dokumenter som er nært forbundet er plassert i nærheten av hverandre. Singulærverdi-dekomposisjonen tillater det semantiske "rommet" å reflektere de store mønstrene i dataene og ignorere de mindre viktige påvirkningene. Som et resultat kan dokumenter som inneholder ord som ikke finnes i et gitt dokument fremdeles kunne ende opp i nærheten av hverandre, hvis dette reflekterer de store mønstrene(Deerwester et al., 1990).

I SVD, vil en n-dimensjonalt rektangulær matrise bli dekomponert til tre andre matriser. Gitt en matrise X og formelen:

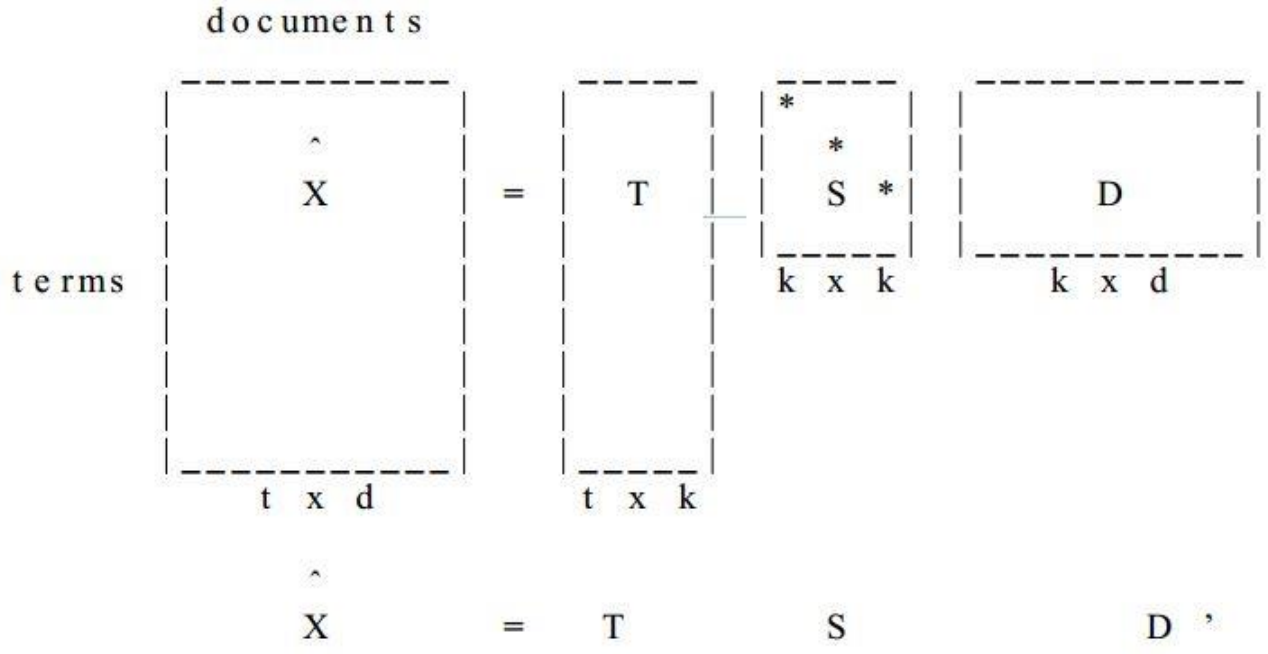
$$X = T_0 S_0 \acute{D}_0.$$

Slik at T_0 og D_0 har ortonormale kolonner og S_0 er diagonal. T_0og D_0 er matriser av venstre og høyre singulærvektorer og S_0 er en diagonal matrise for singulærverdier (Deerwester et al., 1990, s. 11). Figur 2.3 viser et eksempel av singulærverdi-dekomposisjon av en term-frekvens matrise t x d, X, hvor t er antall termer, d er antall dokumenter og m er $\min(t, d)$.



Figur 2.3: Singulærverdi-dekomposisjon(SVD) fra Deerwester et al.(1990, s. 12)

For å generere det "semantiske rommet" som LSI bruker til å finne relaterte dokumenter, må man redusere SVD-matrisene til et k-dimensjonalt rom. Hvor $n \gg k$, n er antall ord i den opprinnelige termfrekvens matrisen og k representerer antall dimensjoner matrisen skal reduseres med. Forskning innen valg av k verdi, viser at det er veldig viktig for å oppnå gode resultater. Dette vil diskuteres senere. Reduksjon av SVD-matrisene til k-dimensjonale rom vises i Figur 2.4.



Figur 2.4: Redusert singulærverdi-dekomposisjon, fra Deerwester et al.(1990, s. 13)

Singulærverdi-dekomposisjonen har blitt redusert til k og $\acute{D}$ har blitt utregnet ved ombytting av rader og kolonner i den opprinnelige matrisen. Det er ved denne reduksjonen LSI lager det "semantiske rommet". Rommet gjør det mulig at dokumenter som inneholder ord som ikke finnes i et gitt dokument fremdeles kan ende opp i nærheten av hverandre. Den opprinnelige matrisen X blir så rekonstruert ved $X = T \times S \times \acute{D}$, men ikke med perfekt nøyaktighet. Det er viktig at det ikke skal være en nøyaktig rekonstruksjon, ettersom det antas at den opprinnelige matrisen var upålitelig. En perfekt rekonstruksjon ville tilsvart den opprinnelige matrisen og da ville det ikke vært noe poeng med singulærverdi-dekomposisjon og reduksjon.

Som nevnt tidligere er det viktig å velge en passende verdi for k, slik at LSI genererer gode resultater. Verdien av k er avhengig av matrisen som skal reduseres, men det er vanlig at verdien er et sted rundt 100 (Deerwester et al., 1990, s. 5). Å velge antall dimensjoner, k, for de reduserte matrisene er en utfordring. En reduksjon i k kan fjerne en stor del av støyen, men å ha for få dimensjoner kan gjøre at man mister viktig informasjon. Som diskutert av Berry, Dumais, & O'Brien (1995, s. 20) kan LSI ytelsen forbedres betraktelig etter en forskjell i k på 10 eller 20 dimensjoner og topper vanligvis mellom 70 og 100 dimensjoner, for å så begynne å avta langsomt.

Den rekonstruerte k-dimensjonale SVD matrisen generert av LSI kan brukes til å beregne likheter mellom vektorene den inneholder ved bruk av cosinus-likhets funksjonen, $sim(d_j, q)$. Funksjonen vil, slik som tidligere, beregne cosinusverdien mellom to vektorer og returnere en verdi, som forteller hvor like de er.

Det er store fordeler med å bruke LSI istedenfor andre klassiske IR gjenfinningssystemer. Disse fordelene er støtten for synonyme og polyseme ord, samt muligheten til å modellere sammenhengen mellom ordene (Rosario, 2000, s. 3). Det er også enkelte ulemper ved LSI, disse er hovedsakelig lagring og effektivitet, men kan også være polyseme ord. På grunn av at hvert ord kun er representert som et punkt i rommet. Det vil si at ord med flere enn en helt annen mening som "form", er representert som den gjennomsnittlige meningen av de forskjellige betydningene. Når den virkelige betydningen er forskjellig fra den gjennomsnittlige betydningen kan LSI faktisk redusere kvaliteten (Deerwester et al., 1990, s. 21).

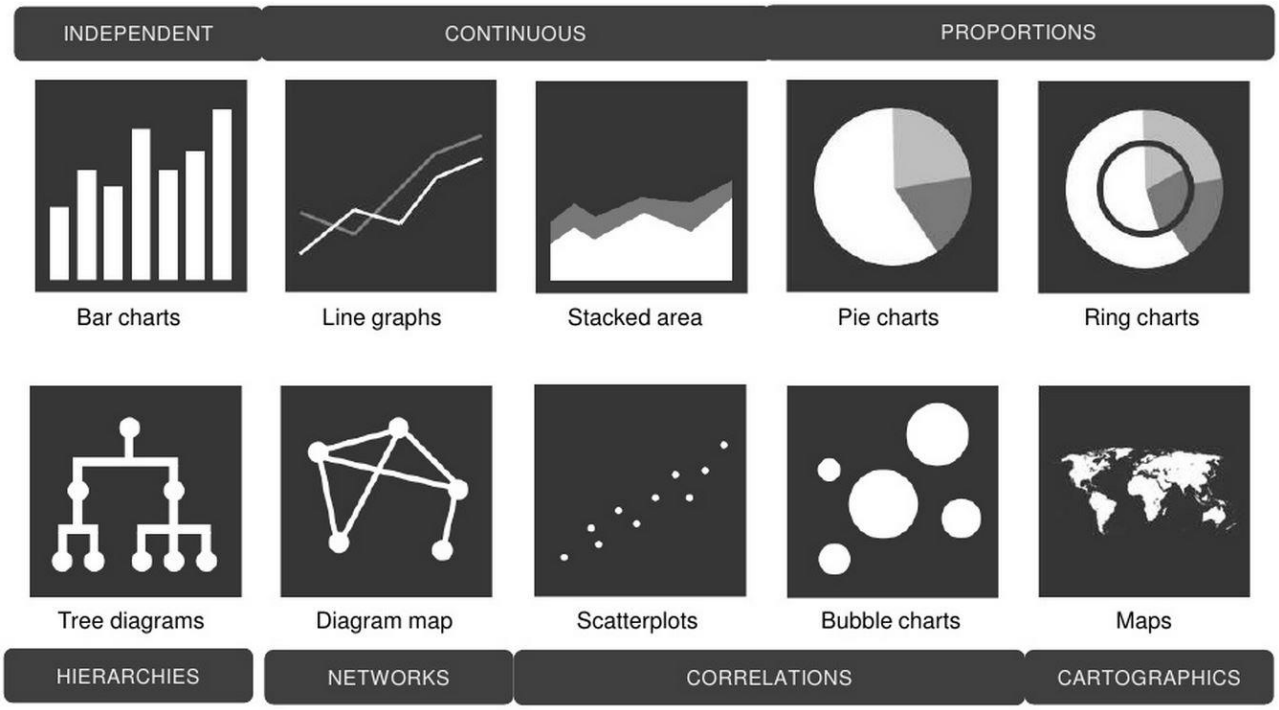
Til syvende og sist, for å bestemme om fordelene oppveier ulempene, er det nødvendig å se på kvaliteten og nøyaktigheten til de forskjellige gjenfinningsmetodene i en gitt kolleksjon. Deerwester et al.(1990) har presentert noen lovende resultater, men har ikke vist at bruken av LSI er overlegen i forhold til den klassiske vektormodellen.

Det blir derfor utført en evaluering av gjenfinningsmetodene LSI og TF-IDF i denne avhandlingen for å analysere hvordan metodene presterer i systemet.

2.2 Datavisualisering

Hovedmålet med datavisualisering er å formidle informasjon effektivt og klart. Man gjennomgår en prosess hvor man konverterer data fra komplekse datasett til visuelle representasjoner som skal presenteres. Ved å presentere data på en visuell form lar man mennesket få innsikt i dataene og trekke konklusjoner, ved å direkte samhandle med dataene. Praktisert med ærlighet og forsiktighet, kan prosessen med visualisering presentere verden på en ny måte, og muliggjør oppdagelse av uventede mønstre og trender i den ellers skjulte informasjonen rundt oss. Murray (2013, s. 1) skriver følgende om datavisualisering: “At its best, data visualization is expert storytelling”.

For at mennesket skal kunne dra nytte av datavisualisering må både estetikk og funksjonalitet gå hånd i hånd. Det vil si at datavisualisering ikke trenger å se kjedelig ut for å være funksjonelle, eller se kunstnerisk ut for å være estetisk. For å formidle informasjon på en effektiv og klar måte, er man derfor nødt til å balansere mellom design og funksjon. Edward Tufte (2001, s. 177), en av de viktigste personene innen fagområdet, beskriver godt design som ”Graphical elegance is often found in simplicity of design and complexity of data.”. Det å finne denne balansen er en tidskrevende prosess som innebærer testing av forskjellige datasett opp mot forskjellige visualiseringer. En kjent fallgruve er å produsere vakre datavisualiseringer som ikke klarer å tjene sitt hovedmål, nemlig å formidle informasjon (Friedman, 2008).



Figur 2.5: Datavisualisering mønstre, fra Kurniawan (2009, s. 29)

Figur 2.5 viser noen grafiske elementer som ofte blir brukt til å presentere data. Det som er viktig ved valg av grafiske elementer er at elementet skal kunne klare å kommunisere det man ønsker på en enkel og intuitiv måte. For å få et innblikk i hvilke grafiske elementer som er brukt for å presentere forskjellige typer data og relasjoner er forskning gjort innen datavisualisering av musikkolleksjoner vært inspirerende for avhandlingen.

Forskning innen datavisualisering av musikkolleksjoner er vanligvis utviklet for å fremheve en egenskap av likhet, som automatisk kan beregnes fra lydsignalene, eller basert på brukergenerte metadata, eller en kombinasjon av begge deler. Forskjellige tilnærminger varierer i hvordan de presenterer data og kriterier for likhet.
Islands of Music prosjektet av Pampalk (2003) genererer en visualisering av musikksamlinger basert på en metafor for geografiske kart. Prosjektet tar for seg utfordringer knyttet til automatisk oppretting av visualiseringer, basert på sanger sin Mp3-fil. Visualiseringen viser sanger som øyer med fjell og åser som reflekterer deres likhet. 1200 egenskaper som gjenspeiler mennesket sitt hørselssystemet er hentet fra sangene sine MP3-filer, og senere redusert til 80 dimensjoner med Principal Component Analysis. Disse blir så presentert i et ”Self-organizing Map” for å generere metaforen for et geografisk kart. Prosjektet lager visualiseringen automatisk og baserer seg kun på egenskaper hentet direkte fra lydfilene, og kan derfor kun støtte identifisering av relasjoner som kan fanges fra lydsignal. Prosjektet kan derfor ikke generere unike visualiseringer for en gitt

musikkolleksjon. Sett bort i fra dette er Islands of Music prosjektet sin datavisualisering imponerende og inspirerende.

Visualiseringer av musikkolleksjoner kan også bruke metadata, enten som et supplement til andre egenskaper, eller frittstående. MusicalNodes prosjektet av Dalhuijsen (2011) bruker grafer til å visualisere album, organisert i henhold til deres brukergenererte sjanger. De interaktive visualiseringene produsert av Torrens, Hertzog, & Arcos (2004) organiserer sanger i henhold til flere dimensjoner av metadata, som sjanger, artist , år, osv. Å bruke metadata i visualiseringer er veldig relevant for denne avhandlingen, men å kun benytte seg av brukergenererte metadata har sine begrensninger, som feil, inkonsekvenser og manglende informasjon. Å kun benytte seg av slike data kan påvirke kvaliteten av visualiseringene. Det vil derfor ikke benyttes eksterne APIer som Last.fm for å hente brukergenererte metadata i denne avhandlingen.

Studiet Visualizing a hit av Tom Enghelhardt og Shaun Ellis(2011) har vært en stor inspirasjon til denne avhandlingen. Studiet tok utgangspunkt i følgende problemstilling «Kan visualisering av 50 år med amerikanske hit sanger hjelpe oss å oppdage trender?» (Ellis & Engelhardt, 2011). For å kunne svare på dette samlet de inn alle sanger fra Billboard Hot 100 fra 1960-2010 og benyttet Echonest APIet for å hente akustiske egenskaper for sangene. Disse akustiske egenskapene var sangens tempo, varighet, musikalsk nøkkel, samt noen abstrakte notasjoner som «energy» og «danceability». Etter datainnsamling hadde de innsamlet data om 4, 200 sanger. Med slike store mengder data vil det vanskelig gjøres å finne mønstre, ettersom man kun ser rader av tall og tekst i databasen. Derfor er datasettet et godt eksempel på nødvendigheten av å visualisere data. Datasettet blir visualisert i Google Motion Chart og Tableau, slik at det er mulig å se endringer og mønstre over tid eller andre tilnærminger. Ved å bruke flere tilnærminger å se datasettet på, for eksempel ved å sortere etter tempo, år eller varighet, tillates brukere å manipulere datasettet til meningsfulle visualiseringer. Dette åpner for at brukeren selv kan velge hvilke egenskaper ved sangene han ønsker å se på.
Prosjektet benytter seg av metoder innen datainnsamling, lagring og visualisering som har vært inspirerende for denne avhandlingen. Det er derimot flere av deres innsynsvinkler og verktøy det finnes alternative løsninger til, som jeg mener kan føre til et bedre sluttresultat med mer fokus på brukervennlighet og tilgjengelighet. Visualizing a hit prosjektet benytter seg blant annet av predefinerte datavisualiseringer levert av Google Motion Charts og Tableu. Det vil si at visualiseringene ikke nødvendigvis kommuniserer de viktigste punktene i datasettet, ettersom visualiseringene er laget av andre. Visualisering i Tableau fører også til at brukeren er nødt til å laste ned et tilleggsverktøy for å se visualiseringen. Ved å velge en alternativ løsning til verktøy for

visualisering mener jeg at applikasjonen ville hatt bedre brukervennlighet og tilgjengelighet. Slike forbedringer er blitt implementert i denne avhandlingen.

Visualiseringer i denne avhandlingen er basert på og inspirert av forskningen presentert ovenfor. Visualiseringene produsert er bygget på akustiske egenskaper, sangtekster, metadata og det blir brukt to mål for likhet mellom sanger, akustiske egenskaper og sangtekster. For å ha kommet frem til visualiseringer som kommuniserer det som er ønskelig på en enkel og intuitiv måte er det utprøvd flere visuelle representasjoner. De visuelle elementene som ble implementert i det ferdige systemet er hovedsakelig force-directed-graph, kulegraf, linjediagram, ordsky og spredningsplott. Hvorfor akkurat disse elementene er blitt brukt og hvordan de visuelt ser ut presenteres i design og utvikling av Datavisualisering, i seksjon 4.10.

3 Forskningsmetodikk

Forskningsmetodikken som benyttes i denne avhandlingen er hentet fra “Design Science in Information Systems Research” av Hevner, March, Park, & Ram (2004). Hevner et al. (2004) beskriver et konseptuelt rammeverk for å sikre gjennomføring og evaluering av god design-vitenskapelig teknologi basert forskning. Ved design-vitenskapelig forskning menes det forskning som søker å skape innovasjoner som definerer ideer, praksiser, tekniske evner, og produkter gjennom analyse, design, implementering, forvaltning og bruk av informasjonssystemer (Hevner et al., 2004, s. 76).

Retningslinjene gitt i Hevner et al. (2004, s. 83) er:

1. Design som en artefakt
2. Problem relevansen
3. Design evaluering
4. Forskningsbidrag
5. Rigid forskning
6. Design som en søkeprosess
7. Kommunikasjon av forskning

Den første retningslinjen sier at et design-forskningsprosjekt skal produsere en artefakt. I den forstand at artefakter i teknologi basert forskning sjeldent er komplette informasjonssystemer som blir brukt. I stedet er det innovasjoner som definerer ideer, praksis eller tekniske evner. Denne avhandlingen fokuserer på utvikling og testing av en prototype av et system som støtter informasjonsgjenfinning og visualisering av norsk popmusikk. Utviklingsprosessen er nøye beskrevet i kapittel 4

Problem relevansen beskriver målet med forskningen, som er å utvikle en teknologi basert løsning til et viktig og relevant virksomhets problem innen informasjonsgjenfinning og visualisering av norsk popmusikk. Problem relevansen er beskrevet i introduksjonen av avhandlingen og finnes i kapittel 1.

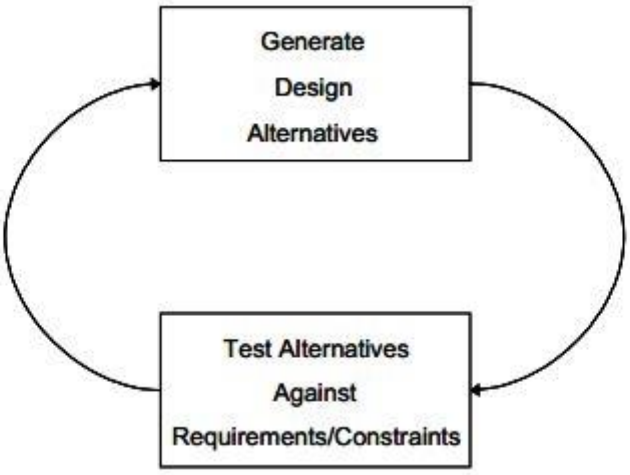
Det tredje punktet understreker viktigheten av en rigid evaluering av den utviklede artefaktet. Hevner et al. (2004, s. 85) forklarer at effekt og kvalitet av et artefakt må være strengt demonstrert via vel utførte evalueringsmetoder. For å vurdere prosjektet er det blitt utført to evalueringer. Evaluering av gjenfinningsmetodene LSI og TF-IDF. Hvor evalueringen resulterte i hvilken gjenfinningsmetode som presterte best i systemet. Det er også utført en evaluering av systemets web-applikasjon og dens evne til å presentere data til bruker, på en forståelig og fortrolig måte. Dette er gjort ved å gjennomføre en usability-test, hvor menneskelig-bruk av systemet sin web-applikasjon i et kontrollert miljø er observert og en System Usability Scale-spørreundersøkelser er analysert. Utformingen og resultater for evalueringene er presentert i kapittel 5.

Det viktigste forskningsbidraget i dette prosjektet er den utviklede prototypen i seg selv. Som vil bidra til å utforske norsk popmusikk og å finne nye sanger. Den ferdige prototypen kan bli brukt som et utgangspunkt for datainnsamling og sammenligning av store mengder data, eller som en inspirasjon til hvordan man kan lage datavisualiseringer av musikkolleksjoner.

Den femte retningslinjen poengterer at forskning skal baseres på rigide og veltestet metoder for både konstruksjon og evaluering av artefaktet. Konstruksjonen av artefaktet bygger på forskning innen informasjonsgjenfinnings-metoder og datavisualiseringer av musikkolleksjoner, presentert i kapittel 2. Evaluering av systemet sin web-applikasjon baserer seg på usability-testing og metode for evaluering av informasjonssystemer som har rot i forskning, presentert i kapittel 5.

Design som en søkeprosess vil si at man bør utforske de mulige implementasjoner av artefaktet ved å iterere gjennom faser hvor man genererer prototyper og tester disse opp mot kravene for prosjektet, se Figur 3.1. Hevner et al.(2004, s. 88) presiserer at å reagere til endringer underveis i prosjektet vil resultere i at artefaktet sin utforming blir mer relevant og verdifull. For å realisere dette, var valg av utviklingsmetodikk svært viktig. Slik at det støtter retningslinjen “design som en

søkeprosess". Valg av utviklingsmetodikk er beskrevet i seksjon 4.1.



Figur 3.1: Generer/test-syklen, fra Hevner et al.(2004, s. 89)

Den syvende og siste retningslinjen foreslått av Hevner et al. (2004) presiserer at kommunikasjon av forskningsresultater skal være rettet mot to typer publikum, teknologi-orienterte og administrasjon-orienterte. Opprinnelig er denne retningslinjen rettet mot det kommersielle, men passer også godt inn i det akademiske. Ettersom akademisk publikum kan også ha forskjellig forståelse for teknologi. Derfor er resultatene av denne avhandlingen presenteres med ulike nivåer av tekniske detaljer, slik at teknologi-orienterte kan ta i bruk artefaktet, samtidig som administrasjon-orienterte kan forstå bruksmåten til artefaktet. For å oppnå dette har det vært fokus på å prøve å introdusere komplekse algoritmer og metoder slik at begge typer publikum kan forstå det.

I tillegg er arbeidet publisert på Internett, anderslangseth.no[2], slik at musikkelskere verden rundt kan utforske norsk popmusikk. Ved å gjøre systemet tilgjengelig på Internett er det mulig for andre å bruke systemet til det de selv finner interessant og muligens kontakte meg gjennom Internett for å kunne bidra til å forbedre systemet.

[2] http://anderslangseth.no/VgListaViz/app/#/

Oversikt over retningslinjene.

Retningslinjer	Kapittel/Seksjon
1. Design som en artefakt	Kapittel 4
2. Problem relevansen	Kapittel 1
3. Design evaluering	Kapittel 5
4. Forskningsbidrag	Systemet i seg selv
5. Rigid forskning	Kapittel 2 og 5
6. Design som en søkeprosess	Seksjon 4.1
7. Kommunikasjon av forskning	Hele avhandlingen

4 Design og utvikling

Dette kapittelet beskriver utviklingsmetodikken, utviklingsprosessen, og hvilke komponenter av systemet som ble utviklet i de forskjellige stadiene av utviklingen. Det vil også presenteres en oversikt av systemet sine komponenter, for å så gå i dybden på de viktigste delene. Videre i kapittelet beskrives verktøyene ettersom de er blitt brukt, men de større rammeverkene presenteres til slutt.

4.1 Iterativ og inkrementell utviklingsmetodikk

Utviklingsmetodikken brukt i avhandlingen bygger på å utvikle et system i flere iterasjoner(iterativt) og i mindre komponenter(inkrementelt) om gangen. Dette gjør det mulig å fokusere på å ferdigstille en iterasjon/komponent av gangen. I hver iterasjon blir et komponent designet, implementert, testet og er ikke ferdig før det er et kjørbart/fungerende komponent (Larman, 2004, s. 14). Hver iterasjon kan også inneholde en eller flere endringer, implementasjoner eller tester. Dette gir mer fleksibilitet for endringer i en iterasjon.

Grunnet at krav og utforming av systemet var blitt gjort i et tidlig stadiet i utviklingen, så var det alltids en mulighet at alternative løsninger og ideer dukket opp underveis. Det har derfor vært viktig å kunne reagere på endringer. Ved å arbeide iterativt, har systemet blitt implementert kontinuerlig og hver iterasjon har blitt evaluert for å finne ut hvilke endringer som er nødvendige for å produsere

et tilfredsstillende sluttprodukt. Antall iterasjoner nødvendig for å produsere et fungerende system er ikke et statisk nummer og har vært avhengig av systemets størrelse, samt potensielle endringer som har oppstått i løpet av utviklingsprosessen. Systemet ble til slutt utviklet over seks iterasjoner. Hvor de første iterasjonene, før implementasjon, ble brukt til å identifisere prosjektets visjon, krav (funksjonelle og ikke-funksjonelle) og arkitektur.

4.2 De forskjellige fasene av utviklingen

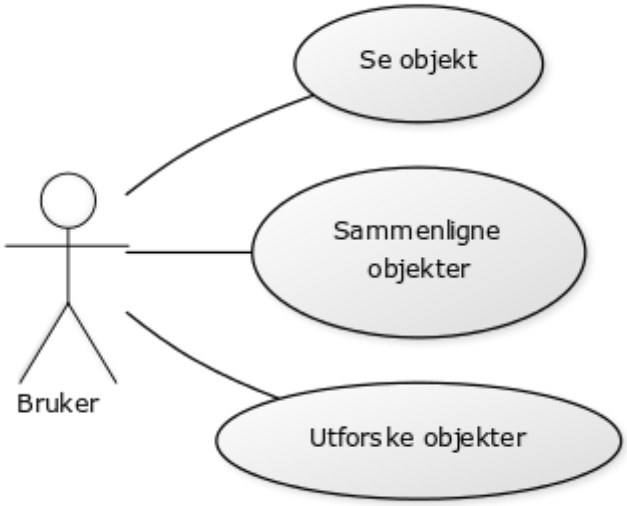
Denne seksjonen inneholder en oversikt over hvilke iterasjoner som ble gjennomført i utviklingen av systemet og i hvilken rekkefølge. Oversikten over iterasjonene er beskrevet på et høyt nivå, for å kunne gi en kort forklaring av utviklingsprosessen som de fleste vil forstå. Videre i de neste seksjonene er implementasjonen av forskjellige komponenter beskrevet på et detaljert nivå med kodesnutter og teknologiske begrunnelser for enkelte avgjørelser.

4.2.1 **Iterasjon 1**

Den første iterasjonen startet med å identifisere prosjektets visjon, omfang og krav (funksjonelle og ikke-funksjonelle). Prosjektets visjon er som tidligere nevnt i introduksjonen, å utvikle et system med støtte for informasjonsgjenfinning og visualisering av relasjoner i norsk popmusikk. For å kunne oppnå visjonen ble det definert følgende funksjonelle krav som definerer spesifikke bruker handlinger i form av use-caser:

Use-casene er designet som “casual user case” ifra Cockburn (2001, s. 121) sine dimensjoner av use-caser. Det vil si at utformingen av en usecase er mer fleksibel og ikke følger strenge standarder innen formattering og innhold.

Figur 4.1 viser use-casene til en bruker i systemet. Objekt representerer artist, liste eller sang.



Figur 4.1: Use case illustrasjon

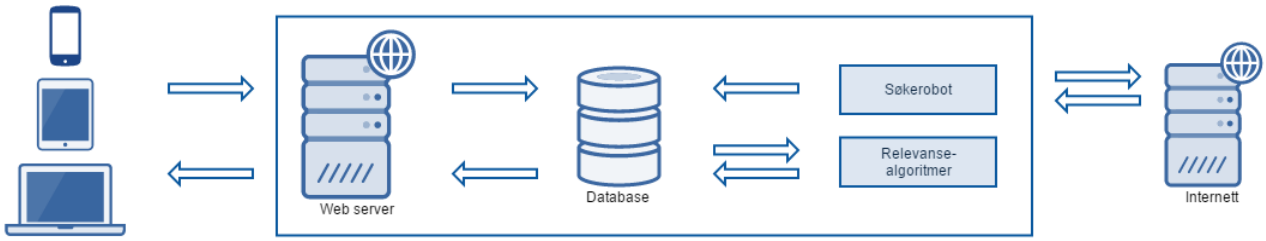
Use case: Se objekt	Forklaring: Bruker ønsker å bli presentert med informasjon om et gitt objekt(artist, liste, sang). Handling: Brukeren klikker på et objekt eller søker opp et bestemt objekt. Resultat: Brukeren blir presentert med informasjon om objektet.
Use case: Sammenligne objekter	Forklaring: Bruker ønsker å bli presentert med informasjon om likheten mellom to objekter(artist, liste, sang). Handling: Brukeren har valgt seg ut to objekter han ønsker å sammenligne. Resultat: Brukeren blir presentert med informasjon om de to objektene satt opp mot hverandre.
Use case: Utforske objekter	Forklaring: Bruker ønsker å utforske forskjellige typer objekter(artist, liste, sang). Handling: Brukeren har valgt ut hvilke type objekter han ønsker å utforske. Brukeren endrer egenskaper eller filtre for objektene han ønsker å se. Resultat: Brukeren blir presentert med objekter på bakgrunn av sine valg av egenskaper og filtre.

De ikke-funksjonelle kravene angir kriterier som brukes for å bedømme bruken av systemet:

- Brukervennlighet er et kriteriet for å sikre at brukeren enkelt skal kunne bruke systemet. Brukervennligheten er viktig for at en bruker skal forstå datavisualiseringene, klare å navigere rundt i systemet og ha en god opplevelse når han bruker det.
- Ytelse, slik at brukeren ikke skal trenge å vente lenge på søkeresultater, visning av data eller generell interaksjon.
- Tilgjengelighet. I denne settingen vil det si at brukeren skal ha muligheten til å bruke systemet uavhengig av lokasjon, altså at det skal være tilgjengelig på Internett.
- Åpen kildekode, for å kunne bygge systemet på andre åpen kildekode prosjekter, men også for å fremme bruken av åpen kildekode.

4.2.2 **Iterasjon 2**

På bakgrunn av systemets visjon ble det utarbeidet en systemarkitektur som vil opprettholde de forskjellige kravene listet ovenfor. Systemarkitekturen er illustrert i Figur 4.2.



Figur 4.2: Illustrasjon av systemarkitektur

Som illustrert i Figur 4.2 består systemet av følgende komponenter:

Web-serveren som håndterer forespørsler fra bruker og returnerer HTML-sider. Denne løsningen tar for seg kravet om tilgjengelighet, ettersom Internett er så og si tilgjengelig overalt. Web-serveren vil også bestå av et API som gjør det mulig å kommunisere med databasen, slik at HTML-sidene vil inneholde relevante data til brukeren sin forespørsel.

Databasen oppretter, oppdaterer og står for tilgang til all data som vil bli brukt i systemet. Dette vil være data om norsk popmusikk, slik som artister, lister og sanger, med tilhørende akustiske egenskaper og lyriske data.

Søkeroboten vil bli brukt for datainnsamling fra Internett via nettsider og APIer. Dette innebærer å samle inn data fra alle listene som er tilgjengelig fra VG-Lista sine nettsider. Videre skal akustisk data og sangtekster hentes for alle sanger i fra eksterne APIer. Når all data er innsamlet vil det lagres i databasen.

Relevanse-algoritmer/gjenfinningsmetoder vil være det komponenten som kalkulerer "likheten" mellom sanger. Komponenten vil innhente de nødvendige data for å gjennomføre kalkuleringene ifra databasen, utføre kalkuleringene, for å så lagre resultatene i databasen. Ved å gjennomføre denne prosessen i back-end, vil web-serveren og klienten slippe disse tunge kalkuleringene. Det vil føre til bedre ytelse for brukeren, ettersom web-serveren kun trenger å hente data ifra databasen.

For å kunne realisere denne system-arkitekturen ble det valgt passende verktøy og teknologier, som alle er åpen kildekode, slik at det er mulig for andre utviklere å realisere et likt type prosjekt.

4.2.3 **Iterasjon 3**

I den tredje iterasjonen startet implementasjonen av systemet. Denne iterasjonen inneholder implementasjon av søkeroboten og databasen.
Innsamling av data fra ca. 3 000 lister og ca. 60 000 sanger i fra VG-Lista sine nettsider er en endeløs tidskrevende prosess hvis det skal gjøres manuelt. Denne prosessen ble derfor automatisert ved å implementere en søkerobot for å webskrape nettsider og eksterne APIer på en effektivt og rask måte. Implementasjonen av søkeroboten er beskrevet nøyere i seksjon 4.4.
Søkeroboten gikk gjennom flere faser av datainnsamling for å hente all nødvendig data for systemet:

- Innsamling av data om lister og sanger fra VG-Lista.
- Innsamling av akustiske egenskaper for sanger.
- Innsamling av sangtekster(lyrikk) for sanger.

Ettersom søkeroboten innhentet sangtekster, som senere vil bli brukt i gjenfinningsmetodene ble denne iterasjonen også brukt til å implementere en stoppordsliste. Dette var for å fjerne ord som ikke er av betydning for gjenfinningsmetodene. Implementasjonen er i seksjon 4.5.

All innsamlet data ble så lagret i databasen for senere bruk. Gitt den store mengden av data som håndteres, ble det brukt mye tid på å vurdere ulike databaser for lagring av innhentede data. Gitt at

resten av prosjektet er bygget rundt hvordan data blir behandlet, hentet og lagret, virket det nødvendig å legge særlig vekt på effektivitet for å hindre en potensiell flaskehals i denne viktige delen.

4.2.4 **Iterasjon 4**

Frem til dette stadiet var det kun lagret data om lister og sanger, hvor kun sanger inneholdt interessante data som akustiske egenskaper og sangtekster. For å kunne presentere et "større" bilde av norsk popmusikk, ble data lagret i sangene brukt til å generere nye data. Slik som akustisk data og sangtekst for et gitt år i norsk popmusikk. Prosessen for å generere disse dataene inneholdt hovedsakelig å finne gjennomsnittet av data fra alle sanger ifra det gitte året. Det ble også generert data for artister og lister. Prosessen av å generere dataene er beskrevet grundigere seksjon 4.6.

4.2.5 **Iterasjon 5**

I den femte iterasjonen ble gjenfinningsmetodene TF-IDF og Latent Semantic Indexing implementert. Metodene tar hovedsakelig to sanger og returnerer en verdi som tilsvarer hvor like de to sangtekstene er. Grunnen til at det ble implementert to forskjellige gjenfinningsmetoder var på grunn av deres forskjellig tilnærming til sammenligning av dokumenter(sangtekster). Den tekniske implementasjonen av TF-IDF blir presentert i seksjon 4.8, Latent Semantic Indexing i seksjon 4.9 og evalueringen av de blir gjennomgått i seksjon 5.1.

Videre i iterasjonen ble det implementert et mål for likhet mellom sanger, basert på akustiske egenskaper. Sammenligningsalgoritmen implementert for å komme opp med målet for likhet baserer seg hovedsakelig på sammenligning av numeriske verdier. Implementasjonen beskrives i seksjon 4.7.

Til slutt ble den best presterende gjenfinningsmetoden brukt til å finne relasjoner mellom sanger på et lyrisk nivå og sammenligningsalgoritmen for akustiske egenskaper på et akustisk nivå. Resultatene ble så lagret i databasen, slik at de senere kunne brukes i visualiseringer.

4.2.6 **Iterasjon 6**

Det neste steget i implementasjonen av systemet bestod av å presentere de interessante data som var blitt innsamlet og generert. For at visualiseringene skulle være tilgjengelige for de fleste og ikke kreve tilleggsverktøy, ble de presentert i en web-applikasjon. Web-applikasjonen ble implementert med følgende HTML-sider:

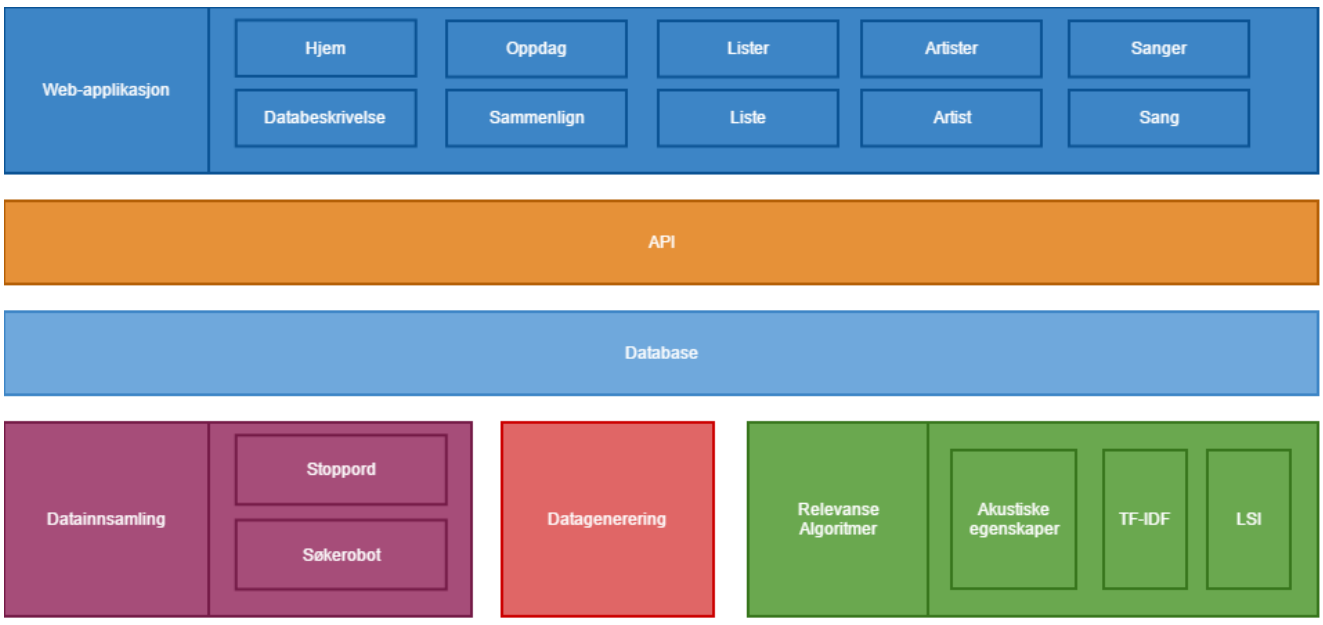
- Hjem (Appendiks A Startside)
    - Presenterer nøkkeldata om prosjektet, antall sanger, artister, osv.
    - Presenterer alle listene fra VG-Lista i en dynamisk tabell.
- Databeskrivelse (Appendiks B Databeskrivelse)
    - Presenterer de forskjellig typene akustiske egenskaper som brukes i prosjektet, slik at brukere kan få en introduksjon til hva de betyr.
- Lister (Figur 4.11)
    - Presenterer endringer i de akustiske egenskapene over listenes år i et dynamisk linjediagram, hvor man kan filtrere etter akustiske egenskaper.
- Artister (Appendiks D Artister)
    - Presenterer artister ut ifra de akustiske egenskapene i et dynamisk stolpediagram, hvor man kan filtrere etter akustiske egenskaper.
- Sanger (Appendiks F Sanger)
    - Presenterer sanger ut ifra akustiske egenskaper i et dynamisk stolpediagram, hvor man kan filtrere etter akustiske egenskaper.
- Oppdag (Figur 4.14)
    - Presenterer et spredningsplott av det brukeren ønsker å oppdage, enten det er artist, liste eller sang, med mulighet til å filtrere etter akustiske egenskaper og år.
- Sammenlign (Figur 4.15)
    - Presenterer to objekter(artist,liste,sang) i forhold til hverandre, ut ifra akustiske egenskaper og sangtekst/lyrikk illustrert i en ordsky.
- Liste (Appendiks C Liste)
    - Presenterer data om listen, slik som akustiske egenskaper og lyrikk.
- Artist (Appendiks E Artist)
    - Presenterer data om artisten, slik som akustiske egenskaper, sangtekster, plasseringer på listene og akustiske egenskaper sin endring i artisten sine sanger.

- Sang (Figur 4.12 & Figur 4.13)
    - Presenterer data om sangen, slik som akustiske egenskaper, sangtekst og plasseringer på listene.

Implementasjon og diskusjon om de viktigste datavisualiseringene for systemet er nøye presentert i seksjon 4.10 og 4.11.
For å teste web-applikasjonen og de implementerte nettsidene er det blitt utført en usability-test. Hvor testdeltakere har utført forskjellige oppgaver, mens de ble observert. Diskusjon av usability-test resultatene er i seksjon 5.2.4.

4.3 Systemoversikt

Systemet fulgte systemarkitekturen presentert i Figur 4.2 og har gjennom de forskjellige iterasjonene presentert i seksjon 4.2 blitt videre delt inn i flere løst koblede komponenter, som fungerer uavhengig av hverandre. At de er løst koblede har vært en stor fordel i utviklingsprosessen, ettersom det har vært enkelt å ferdigstille ett komponent av gangen.



Figur 4.3: Komponent oversikt

Figur 4.3 presenterer en oversikt over følgende komponenter implementert:

- Komponent for søkerobot
- Komponent for å fjerne stoppord
- Komponent for å generere nye data, slik som data om artist og år.
- Komponent for gjenfinningsmetoden TF-IDF
- Komponent for gjenfinningsmetoden Latent Semantic Indexing(LSI)
- Komponent for å sammenligne objekter basert på akustiske egenskaper.
- Web-applikasjonen, bestående av alle HTML-sider.

Videre vil implementasjon av disse komponentene beskrives i rekkefølgen de ble implementert i og på et mer teknisk detaljert nivå.

4.4 Søkerobot

Som nevnt tidligere defineres norsk popmusikk i dette prosjektet som sanger på VG-Lista Topp 20 gjennom årene. Listene er tilgjengelig på VG Lista sine hjemmesider[3] og inneholder ca. 3,000 lister og 60,000 sanger. I første steg i datainnsamlings-prosessen skulle data om alle lister og sanger innsamles. Videre skulle akustiske egenskaper og lyrisk data for hver sang hentes. Innhenting av så mye data var en utfordring, ettersom det er en endeløs tidskrevende prosess hvis det skulle gjøres manuelt. Derfor var det ønskelig å effektivisere denne prosessen. En løsning for å effektivisere datainnsamlingen var å implementere en søkerobot for innhenting av data fra VGLista og fra eksterne applikasjonsprogrammeringsgrensesnitt(API). Søkeroboten bidro til effektivitet og minsket muligheten for menneskelige feil som kunne oppstått ved å ”klippe og lime”.

Søkeroboten er implementert med Jsoup sitt Java-bibliotek. Jsoup[4] er et Java-bibliotek for arbeid med «real-world» HTML. Det har et veldig praktisk API for å hente ut og manipulere data, samt bruker det beste av DOM, CSS, og jQuery-lignende metoder til å finne elementer. Biblioteket var enkelt å komme i gang med.

[3] http://lista.vg.no/
[4] http://jsoup.org/

For å samle inn all nødvendig data gikk søkeroboten gjennom flere faser av webskraping:

- Innsamling av data om lister og sanger fra VG-Lista.
- Innsamling av akustisk data for sanger fra Echonest APIet.
- Innsamling av sangtekster(lyrikk) for sanger fra Musixmatch APIet.

Videre beskrives disse fasene og hvordan søkeroboten ble implementert for å håndtere de forskjellige fasene.

4.4.1 Innsamling av data fra VG-Lista

For å innsamle data fra VG-Lista startet søkeroboten opp med en dynamisk URL som itererte gjennom alle lister over uker og år i VG-Lista Topp 20 listene. Den dynamiske webadressen(URL) til en liste på VG-Lista sine nettsider er:

http://lista.vg.no/liste/topp-20-single/1/dato/Aar/uke/Uke.

Hvor variabel Aar representerer året for listen og Uke er listen sin uke. For hver URL søkeroboten besøkte utforsket den nettsiden og hentet ut ønskelige data fra listen. Kodesnutt 4.1 viser hvordan søkeroboten besøkte alle listene fra 1960 til og med 2014.

```
for(int aar=1960; aar <= 2015; aar++){
      for(int uke=1; uke <= 53; uke++){
            String url = "http://lista.vg.no/liste/topp-20-
            single/1/dato/" + aar +"/uke/" + uke;
            crawler.getList(url);
      }
```

Kodesnutt 4.1: Datainnsamling med søkerobot

Søkeroboten utfører en forespørsel til web-serveren og får igjen en HTML-side. Ut ifra HTML-siden må søkeroboten gjenkjenne forskjellige HTML-elementer og mønstre på bakgrunn av regler den er blitt gitt. Dette er for å sikre at den parser ønskelig data på korrekt måte. Eksempler på regler man kan gi en søkerobot, er at den skal hente ut data som finnes i et paragraf-element og parse dette som en Java-Integer, slik at det senere kan utføres numeriske operasjoner med det. Videre er et reelt

eksempel fra når søkeroboten besøker VG-Lista Topp 20, Uke 15, 2015[5]:

```
▼<div itemscope itemtype="http://schema.org/MusicPlaylist">
   <meta itemprop="numTracks" content="40">
   <h1 class="listHeader list1" itemprop="name">Topp 20 Single uke 15, 2015</h1>
  ▼<table class="chart">
   ▶<thead>…</thead>
   ▼<tbody>
    ▼<tr onclick="location.href='/artist/kygo-feat-parson-james/singel/stole-the-show/6288'" itemprop="albums" itemscope
    itemtype="http://schema.org/MusicAlbum">
     ▶<th scope="row">…</th>
      <td>1</td>
      <td>3</td>
     ▼<td class="left">
        <span class="play">play</span>
        <img src="http://varnish01.music.aspiro.com/im/im?h=200&w=200&albumid=43550578" itemprop="image">
        <a href="/artist/kygo-feat-parson-james/singel/stole-the-show/6288" title="Stole The Show" class="album" itemprop=
        "name">Stole The Show</a>
        <a href="/artist/kygo-feat-parson-james/6267" title="Kygo feat. Parson James" class="artist" itemprop="byArtist">Kygo
        feat. Parson James</a>
      </td>
    </tr>
    ▶<tr onclick="location.href='/artist/matoma-the-notorious-b-i-g-feat-ja-rule-ralph-tresvant/singel/old-thing-back/6280'"
    itemprop="albums" itemscope itemtype="http://schema.org/MusicAlbum">…</tr>
    ▶<tr onclick="location.href='/artist/major-lazer-feat-moe-dj-snake/singel/lean-on/6285'" itemprop="albums" itemscope
    itemtype="http://schema.org/MusicAlbum">…</tr>
    ▶<tr onclick="location.href='/artist/madcon-feat-ray-dalton/singel/don-t-worry/6276'" itemprop="albums" itemscope itemtype=
    "http://schema.org/MusicAlbum">…</tr>
    ▶<tr onclick="location.href='/artist/rihanna-kanye-west-paul-mccartney/singel/fourfiveseconds/6262'" itemprop="albums"
    itemscope itemtype="http://schema.org/MusicAlbum">…</tr>
    ▶<tr onclick="location.href='/artist/lost-frequencies/singel/are-you-with-me/6268'" itemprop="albums" itemscope itemtype=
    "http://schema.org/MusicAlbum">…</tr>
    ▶<tr onclick="location.href='/artist/ellie-goulding/singel/love-me-like-you-do/6258'" itemprop="albums" itemscope itemtype=
    "http://schema.org/MusicAlbum">…</tr>
```

Figur 4.4: HTML-siden til VG Lista Topp 20, uke 15, 2015.

Figur 4.4 viser deler av HTML-siden til listen for uke 15, 2015 i fra VG-Lista Topp 20. Søkeroboten er blitt instruert til å følge følgende regler for parsing av HTML-sider:

- Listen finnes i div-elementet med microdata attributtet itemtype=”MusicPlaylist”.
- H1-elementet i listen inneholder naturlig-tekst, eksempel “Topp 20 Single uke 15, 2015”, hvor ukenummeret er listet etter “uke ”-stringen og året er etter “, “-stringen.
- Table-elementet med class=”chart” i listen inneholder tabellen/listen med alle sangene.
- Hvert tr-element(tabell rad) i tabellen inneholder en sang
    - Første td-element(celle) inneholder sangens posisjon.
    - Andre td-element(celle) inneholder sangens posisjon sist uke.
    - Tredje td-element(celle) inneholder to anchor-elementer med data om sangen.
        - Første anchor-elementet inneholder sangens tittel i title-attributtet
        - Andre anchor-elementet inneholder sangens artist i title-attributtet.

Alle listene, fra 1960-2014 er strukturert slik at søkeroboten kan følge instruksene ovenfor. Data innhentet fra VG-Lista var hele 2,800 lister med 4,500 unike sanger. Neste fase for søkeroboten er å innsamle akustisk egenskaper for alle sangene.

[5] http://lista.vg.no/liste/topp-20-single/1/dato/2015/uke/15

4.4.2 **Innsamling av akustiske data.**

For å samle inn akustiske data for sangene er det blitt brukt et åpent Web API tilgjengeliggjort av EchoNest[6]. EchoNest er et musikk intelligens selskap grunnlagt av to tidligere MIT-studenter og beskriver seg som bransjens ledende musikk analyse selskap, som gir utviklere den dypeste forståelse av musikkinnhold for musikkentusiaster. Selskapet har kunder som BBC, MTV og Spotify. Deres rike data API og utvikler-verktøy blir brukt til å bygge smartere musikkopplevelser som hjelper musikkelskere å oppdage, dele og samhandle med musikken de elsker.

Echonest sitt API har en metode som tar en sang sin artist og tittel, for å så returnere de akustiske dataene tilknyttet sangen i JSON-format. Søkeroboten sender en HTTP GET forespørsel til APIet med sangens tittel og artist i request body. Eksempel på en forespørsel er sangen “Optimist” av “Jahn Teigen” med følgende URL:

http://developer.echonest.com/api/v4/ song/search ?api_key=MYAPIKEY &artist=Jahn%20Teigen&title=Optimist.

Responsen fra APIet er vist i Kodesnutt 4.2.

```
{
  "artist_name": "Jahn Teigen",
  "title": "Optimist",
 "audio_summary": {
    "danceability": 0.667098,
    "duration": 174.37288,
    "energy": 0.815244,
    "key": 2,
    "loudness": -6.207,
    "mode": 1,
    "tempo": 118.088,
    "time_signature": 4
  }
}
```

Kodesnutt 4.2: Respons fra Echonest APIet

[6] http://the.echonest.com/

For å hente ut data fra JSON-filen presentert i Kodesnutt 4.2 er søkeroboten instruert til å parse JSON-egenskapene artist_name, title og hele audio_summary objektet med tilhørende egenskaper. JSON-responsen inneholder akustiske egenskaper til sangen. Disse egenskapene er analysert av Echonest ut ifra sangen sin opprinnelige lyd-fil. Det er også inkludert noen abstrakte begreper som «energi» og «dansbarhet», som er generert gjennom komplekse algoritmer (Ellis & Engelhardt, 2011).Følgende er en beskrivelse av de akustiske egenskapene returnert fra APIet.

- Dansbarhet(Danceability)
  Representerer hvor egnet en sang er å danse til, verdien er et nummer mellom 0 og 1. Det er regnet ut av musikalske elementer som best karakteriserer dansbarhet. Dette er tempo, rytme stabilitet, slag styrke, og samlet regularitet.
- Varighet(Duration)
  Sangens varighet, målt i sekunder.
- Energi(Energy)
  Er et nummer mellom 0 og 1, som representerer hvor energisk en sang er. Typisk for energiske låter er at de er raske, høylytte og bråkete. For eksempel har sjangeren death metal høy energi, mens et forspill av Bach er lavt på skalaen.
- Nøkkel(Key)
  Representerer nøkkelen som EchoNest mener sangen er i. Tonearter starter på 0 (C) og følger den kromatiske skalaen i stigende rekkefølge. I dette tilfelle representerer verdien 1 en nøkkel av en sang i D-dur.
- Lydstyrke(Loudness)
  Den generelle lydstyrken til en sang er målt i desibel(dB). En sang har forskjellig lydstyrke over flere segmenter. Verdien som blir benyttet for lydstyrke er derfor gjennomsnittsverdien målt gjennom hele sangen.
- Skala/Akkord(Mode)
  Representerer om sangen er spilt i moll (0) eller dur (1) nøkkel og bør sees sammen med nøkkelen.
- Tempo(Tempo)
  Er det estimerte tempoet til en sang, målt i taktslag per minutt(BPM). I musikkteori er tempo hastigheten i et gitt stykke og stammer direkte fra den gjennomsnittlige takt varigheten. For eksempel så vil et stykke som går i 60 BPM ha ett taktslag i sekundet.
- Taktart(Time Signature)
  Er den anslåtte og samlede taktarten til en sang. Altså hvor mange slag som inngår i en gitt takt. Eller hvor mange deler en takt er delt opp i.

Å benytte Echonest APIet for tilgang til akustiske data har spart prosjektet for mange timer arbeid med å utvikle et eget komponent for å analysere sanger, men også potensielle opphavsrettslige problemer. Ettersom det er i dag umulig å få tilgang til 4, 500 sanger på en lovlig måte. De dataene EchoNest APIet leverer er ikke et brudd mot loven om opphavsrett, ettersom det er lov å dele akustisk data, men ikke selve lyden. Med tanke på prosjektets tidsaspekt muliggjorde bruken av APIet at det kunne fokuseres mer på andre viktige områder i systemet.

Selve innsamlingen fant akustiske egenskaper for 3, 487 av 4, 500 unike sanger. De fleste sangene som Echonest APIet ikke kunne levere akustiske egenskaper til var eldre "ukjente" sanger, ofte skandinaviske. Et eksempel på en slik sang er "Ungkarsvalsen" av Gunnar Engedahl & Erling Stordahl som var listet i uke 1, i 1960. Dette fører til at data som presenterer mønstre i norsk popmusikk rundt år 1960-1970 ikke er like utfyllende som for de nyere årene, hvor det er registrert akustiske egenskaper for så å si alle sanger.

4.4.3 **Innsamling av sangtekster**

Sangen blir så sendt videre i datainnsamlingsprosessen for å hente ut sangteksten. Innsamling av sangteksten kunne blitt gjort på flere forskjellige måter. Et alternativ var å la søkeroboten søke etter sangtekster på Internett. Dette var derimot en løsning som det var ønskelig å unngå, ettersom de fleste nettsteder som inneholder sangtekster, ikke opprettholder loven om opphavsrett eller unngår den. Sangtekster er lisensiert og er ulovlig å publisere på nett uten samtykke fra artist og/eller plateselskap. For å unngå problemer rundt opphavsrett ble Musixmatch[7] sitt API brukt.

Musixmatch APIet tilbyr mer enn 20 millioner sanger i 50 forskjellige språk fra 1 million artister. Musixmatch sier selv at det er den raskeste, mest kraftige og lovlige måten å vise sangtekster på. Selskapet var såpass interesserte i systemet at de lot meg benytte en kommersiell lisens og uten dette ville ikke systemet vært mulig å gjennomføre på lik måte.

Søkeroboten sender en HTTP GET forespørsel til APIet med sangens tittel og artist i request body. Et eksempel er en forespørsel om sangen "Optimist" av "Jahn Teigen", med følgende URL:

> http://ec2-54-166-28-138.compute-1.amazonaws.com//research/v1.0/bow.get?q=Jahn%20Teigen%20Optimist.

Responsen fra APIet vises i Kodesnutt 4.3.

[7] https://www.musixmatch.com/

```
{
  "artist_name": "Jahn Teigen",
  "track_name": "Optimist",
  "Bag_of_words": [
    ["begynne", 1], ["oss", 2], ["skal", 3], ["Optimist", 10], ["sist", 4], ...
  ]
}
```

Kodesnutt 4.3: Respons fra Musixmatch APIet

Søkeroboten er instruert til å hente ut Bag_of_words arrayet som inneholder alle ordene og verdiene i arrays. Bag_of_words-arrayet returnert fra APIet representerer sangteksten i "bag of words"-modellen, hvor den eksakte rekkefølgen av ordene i dokumentet blir ignorert, men antall forekomster av hvert ord er av betydning. Dermed er teksten "Mary er raskere enn John" i denne modellen, identisk med "John er raskere enn Mary". Selv om dette kan sees på som et problem, så er det likevel intuitivt at to sangtekster med lignende "bag of words"-representasjoner er like i innhold (Manning & Raghavan, 2009, s. 117).
Ordene i bag of words-modellen returnert fra APIet har allerede gjennomgått stemming. Det vil si at ord er redusert til sin grammatiske rot, ved å blant annet fjerne endinger(mennesker -> menneske). Dette reduserer settet av unike ord, ettersom "mennesker" og "mennesket" vil bli representert som samme ord. Å redusere settet av unike ord på denne måten er ønskelig for store samlinger av tekstlige data (Baeza-Yates & Ribeiro-Neto, 2011, s. 63). For å videre redusere settet av unike ord var det av interesse å fjerne stoppord, slik at gjenfinningsmetodene kan prestere så bra som mulig.

Musixmatch APIet fant sangtekster for 2, 745 av 4, 500 unike sanger. Det var ønskelig å finne sangtekster til flere sanger, men 2,745 sangtekster er fremdeles et stort nok datasett til å kunne få interessante resultater fra gjenfinningsmetodene.

4.5 Stoppordliste

Stoppord er ord som ikke bærer mening i naturlig språk, på grunnlag av at de ikke tilføyer signifikant semantikk til dokumentet eller fordi de er veldig repetitive, og kan derfor ignoreres. Eksempler på stoppord som ikke bærer mye mening i det norske språk er: "en", "den", "ved", osv.

Ofte er det akkurat disse ordene som er mest gjengående og derfor finnes flest av i en kolleksjon.

Prosessen ved å fjerne stoppord innebar å sjekke om et gitt ord finnes i stoppordlisten. Hvis det fantes, ble det fjernet fra sangteksten. Utfordringen med å fjerne stoppord var å bygge en stoppordliste som kun fjerner ord som ikke bærer mening i naturlig språk. Dette viste seg å være en utfordring, ettersom de innsamlede sangene inneholdt sangtekster på forskjellige språk, hovedsakelig engelsk, norsk og svensk. Dette førte til at stoppordlisten var nødt til å ta hånd om stoppord fra tre forskjellige språk. Det var ingen enkel problemstilling, ettersom et stoppord i et språk, kan være av betydning i andre språk. For å prøve å håndtere problemet ble alle ord i hele samlingen summert og de 100 mest brukte ordene ble nøye evaluert. Resultatet var nesten kun ord som karakteriseres som stoppord. Fordelingen var flest engelske ord, noen norske og et par svenske.

På grunnlag av disse resultatene ble stoppordlisten prioritert etter språk, i rekkefølgen: engelsk, norsk, svensk. Det vil si at engelske stoppord ble prioritert over norske og svenske. Videre inkluderte stoppordlisten kun stoppord fra norsk og svensk, der hvor de ikke var av betydning i det engelske språket eller inneholdt skandinaviske bokstaver som “æ”, “ø” og “å”. Eksempler på norske ord som ble lagt til i stoppordlisten er “is”, “by” og “and” som er veldig mye brukt og ikke bærer mening på engelsk, men har betydning på norsk. Ord som “jeg”, “deg”, “begge”, “vi”, osv, ble også lagt til etter nøye vurdering av deres mening på engelsk.
Stoppordlisten var også inspirert av ranks.nl[8] sin stoppordliste for engelske ord. I Tabell 4.1 er et utsnitt av de engelske ordene i den implementerte stoppordlisten.

A	About	After	All
Am	An	And	Any
Are	At	Be	Been
But	By	Can’t	Did
Do	Don’t	For	Had
Has	Him	His	I
It	Me	My	No
The	Their	Then	There
Very	Was	We	Why
You	Your	Yours	Yourself

Tabell 4.1: Stoppordliste

[8] http://www.ranks.nl/stopwords

Etter testing opp mot denne stoppordlisten var det fremdeles flere "ubetydelige" ord som gikk igjen i sangtekstene. For å hindre dette, ble de 100 mest brukte ordene evaluert på nytt. Dette førte til at følgende ord ble lagt til i stoppordlisten, se Tabell 4.2:

Of	On	Once	Our
She	So	Some	That
If	In	Into	Is
Under	Until	When	While

Tabell 4.2: Stoppordliste, tillegg

Det var en tidskonsumerende oppgave å utarbeide stoppordlisten, men det var av høy prioritet, ettersom meningsløse ord kan påvirke gjenfinningsmetodene og føre til falske resultater.

4.5.1 **Lagring**

Ved fullført datainnsamling og prosessering av tekst, var det innsamlet data om sangens artist, tittel, sangtekst og akustiske egenskaper for 4, 500 unike sanger, i fra 2, 800 lister. All denne dataen var nødt til å bli lagret for å senere kunne brukes i gjenfinningsmetoder og presentasjon av data i web-applikasjonen. Grunnet den store datamengden er lagring både tids og plass krevende, noe som var nødvendig å ta hensyn til i valg av database. MongoDB ble sett på som en god løsning for databasen. Dette er på grunn av at AngularJS, som er det front-end verktøyet som blir brukt, og MongoDB begge benytter seg av JSON-objekter. MongoDB lagrer JSON-objekter og AngularJS kan enkelt presentere data fra JSON-objekter. Ved å bruke MongoDB som database slapp jeg å ta hensyn til omforming av data mellom database og web-applikasjonen.

En annen fordel av MongoDB er sin struktur, som skiller seg fra den mer statiske strukturen til vanlige SQL eller tabell-baserte databaser. En MongoDB inneholder et sett med kolleksjoner, hvor en kolleksjon inneholder et sett med dokumenter. Et dokument er et JSON-objekt, som har et dynamisk skjema. Dette betyr at dokumenter i den samme kolleksjonen ikke trenger å ha den samme strukturen. Dette gir stor fleksibilitet og er en fordel i systemet. Lagring av sangtekster i SQL ville krevd en tabell med kolonner på størrelse med den sangteksten som inneholder flest unike ord, slik at det er plass til alle ordene. Sangtekster med færre ord, ville da ikke trengt alle feltene og man ville vært nødt til å lagre null-verdier i kolonnene for å fylle de ut. I MongoDB er ikke dette et

problem, grunnet den dynamiske strukturen, som godtar forskjellige data.
På bakgrunn av MongoDB sin struktur er den implementerte databasen delt opp i flere kolleksjoner som holder på dokumenter relevante til sin kolleksjon. Kolleksjonene opprettet ved dette stadiet var "Lists" og "Songs". På bakgrunn av kolleksjonen sitt navn skal det være veldig forståelig å vite hva dokumentene i kolleksjonen omhandler.

Strukturen til dokumentet for en sang ble testet flere ganger, slik at det ville være forståelig og plass-effektivt. Kodesnutt 4.4 viser den endelige strukturen til et dokument for en sang, i dette eksempelet sangen Optimist av Jahn Teigen i kolleksjonen "Songs":

```
{
  "_id": {
      "$oid": "545105a4612026cf3f69b187"
   },
  "artist": "Jahn Teigen",
  "title": "Optimist",
  "audio_summary": {
    "danceability": 0.667098,
    "duration": 174.37288,
    "energy": 0.815244,
    "key": 2,
    "loudness": -6.207,
    "mode": 1,
    "tempo": 118.088,
    "time_signature": 4
  },
  "Bag_of_words": [
    ["begynne", 1], ["oss", 2], ["skal", 3], ["Optimist", 10], ["sist", 4], ...
  ]
}
```

Kodesnutt 4.4: JSON-strukturen til sangen Optimist av Jahn Teigen

Kodesnutt 4.4 representerer et dokument for sangen Optimist av Jahn Teigen. Strukturen er basert på responsene fra de eksterne APIene, EchoNest og Musixmatch, se Kodesnutt 4.2 og Kodesnutt 4.3. Dokumentet inneholder et _id objekt som er en unik verdi auto-generert av MongoDB, slik at alle dokument får en unik identifikator. Videre har dokumentet enkle og beskrivende egenskaper som artist og title, men også mer komplekse lister(arrays) og objekter. Audio_summary objektet holder på de akustiske egenskapene, mens Bag_of_words listen holder på de unike ordene med tilhørende frekvens brukt i sangen. Valg av datatyper er også relevant, slik som at alle akustiske egenskaper sin verdi er av datatypen Number. Dette er for å raskt kunne utføre numeriske operasjoner på disse.

På bakgrunn av den unike identifikatoren _id til et dokument, er det enkelt å lagre dokumenter om lister i en separat kolleksjon, kalt “Lists”, hvor listen refererer til sangene den inneholder. Eksempel på en liste er vist i Kodesnutt 4.5.

```
{
   "_id": {
      "$oid": "545112fb612026cf3f69b2d3"
   },
   "year": 1963,
   "week": 6,
   "list": [
      {"sangRef": "545105a4612026cf3f69b187"},
      {"sangRef": "545105aa612026cf3f69b18a"}
      ...
   ]
}
```

Kodesnutt 4.5: JSON-strukturen til listen for uke 6 i 1963

Kodesnutt 4.5 viser listen fra uke 6 i 1963. Dokumentet er veldig beskrivende og har en liste(array) med navnet “list” som inneholder alle objekter med referanse til alle sangene på listen, sortert etter plassering (1-20) i stigende rekkefølge.

Gitt den store mengden av data som håndteres, var det verdt å vurdere ulike databaser for lagring av innhentede data. Det har blitt tatt hensyn til hvert alternativ sin effektivitet og brukervennlighet. Gitt at resten av prosjektet er bygget rundt hvordan dokument blir behandlet, hentet og lagret, virket det nødvendig å legge særlig vekt på effektivitet for å hindre en potensiell flaskehals i denne viktige delen.

4.6 Generering av data

Frem til dette stadiet var det innsamlet data om lister og sanger, hvor kun sanger inneholdt interessante data som akustiske egenskaper og sangtekster. For å kunne presentere et "større" bilde av norsk popmusikk, måtte data lagret i sangene slåes sammen for å generere nye interessante dokument. Slik at man kan se relasjoner mellom forskjellige artister, lister og år i norsk popmusikk.

Å utføre sammenslåing av alle sanger fra lister for året 1960 er en krevende oppgave å utføre i nettleseren og vil føre til at brukeren er nødt til å vente lenge på at visualiseringer lastes. Derfor var det svært viktig å gjennomføre databehandlingen i back-end, slik at front-end kun trenger å fokusere på interaksjon og presentasjon av data. Valg som dette er gjort på bakgrunn av det ikke-funksjonelle kravet om ytelse som er definert i seksjon 4.2.1.

4.6.1 **Sammenslåing av data**

For å generere akustisk og lyrisk data for artister, lister og år måtte data slåes sammen. Det var en veldig enkel og intuitiv oppgave som for det meste omhandlet å finne gjennomsnittsverdien for de akustiske egenskapene og summere lyrisk data, for alle sangene relatert til dokumentet. Følgende er en kort beskrivelse av fremgangsmåten for generering av de forskjellige dokumentene.

For alle sangene artisten har laget ble akustisk egenskaper sin verdi summert og delt på antall sanger. Dette vil si å summere de akustiske egenskapene som dansbarhet, tempo og varighet, for hver sang. For å så dele verdien på antall sanger artisten har laget, slik at artist dokumentet vil ha akustisk egenskaper basert på gjennomsnittsverdien for alle sanger han har laget i norsk popmusikk.

For å produsere artisten sin lyriske data ble alle ordene og frekvensene til alle sangene han har laget summert. For de ordene som oppstod i flere av sangene ble frekvensen summert, slik at "Bag of words"-modellen ble opprettholdt ved å kun inneholdt unike ord. Artisten vil da inneholde data om

alle ordene han har brukt og frekvensen. Tilleggsdata ble også lagt til. Slik som antall ganger artisten var listet og den beste plasseringen han har hatt på listene.

Liste dokumentene var allerede generert i databasen og hadde de aktuelle sangene i list-arrayet, se Kodesnutt 4.5. Grunnet dette gikk det raskt å samle inn de nødvendige data fra sangene. Det ble generert akustiske egenskaper og lyrisk data ved å finne gjennomsnittet av de akustiske egenskapene og summering av lyrisk data for alle sangene i listen.

For å generere data om et år ble akustiske egenskaper og lyrisk data generert fra alle listene fra et gitt år. Det ble generert data om år fra 1960 og til 2014. Det ble også lagt til tilleggsdata om hvor mange ganger en sang i snitt var listet på VG-Lista Topp 20.

4.6.2 **Lagring**

Lagring av de nye genererte dokumentene fulgte samme praksis som tidligere. Det vil si at dokumentene ble lagret i egne, respektive kolleksjoner, slik at det var enkelt og raskt å finne frem til dokumentene når de skulle presenteres i web-applikasjonen. Det ble derfor laget to nye kolleksjoner, en for hver type dokumenter som er laget. Kolleksjonen “Artists” som holdt på artist dokumentene og kolleksjonen “Years” som holdt på dokumentene om årene.

Selve strukturen på dokumentene var viktig å holde logiske og like i forhold til dokumenter med lik data. Et eksempel er at det nå finnes akustisk egenskaper og lyrisk data i alle dokumenter i alle kolleksjoner. Da er det en fordel at disse dataene er strukturert på lik måte, ettersom det vil gjøre videre behandling og presentasjon av data enklere.

Kodesnutt 4.6 representerer et dokument om artisten “Jahn Teigen”, lagret i kolleksjonen ”Artists”.

```
{
  "_id": {
      "$oid": "54d62bbbe78f4f2e5d9c963e"
   },
  "name": "Jahn Teigen",
  "weekslisted": 75
  "bestplace": 1
  "audio_summary": {
   "danceability": 0.667098, ...
  },
  "Bag_of_words": [
   ["begynne", 1], ...
  ]
}
```


Kodesnutt 4.6: JSON-strukturen til artisten Jahn Teigen

4.7 Algoritme for akustiske egenskaper

For å finne ut om to sanger høres like ut ble det beregnet et mål for likhet mellom sanger på et akustisk nivå. Det ble utprøvd flere enkle algoritmer som enkelt kunne byttes ut ifra forventet resultat. Alle metodene utprøvd sammenlignet to sanger basert på et akustisk likhetskriterie.

Den første metoden innholdt prosentvis sammenligning av to sanger sine akustiske egenskaper, hvor de måtte være minimum 60% like. Det vil si at to sanger kun ble antatt å være like eller å ha en relasjon til hverandre hvis verdiene av alle de akustiske egenskapene var 60% like eller mer. De følgende akustiske egenskapene som ble testet i denne metoden var:

- Dansbarhet(Danceability)

- Varighet(Duration)
- Energi(Energy)
- Nøkkel(Key)
- Lydstyrke(Loudness)
- Tempo(Tempo)
- Taktart(Time Signature)

Når algoritmen ble testet med en delmengde av 50-100 sanger fra databasen, var resultatene uakseptable. Algoritmen viste seg å være for generell og returnerte for mange treff. Ved å manuelt høre på sangene mellom noen beslektetede par, ble det funnet flere par som overhodet ikke var "like". På grunnlag av dette ble algoritmen nøye revurdert og det ble gjort et forsøk på å legge til sangens skala til algoritmen. Den akustiske egenskapen skala er enten moll eller dur og representerer om sangen er "munter" eller "dyster". Å legge til denne boolske verdien i metoden reduserte resultatene drastisk, ettersom boolske verdier enten er like eller ulike.
Videre var det et stort spørsmål om sangens varighet skulle være med i algoritmen. Sangens varighet påvirker ikke om to sanger høres like ut. Varighet er antageligvis mer en komplementær egenskap til sangen. Revurderingen resulterte i å fjerne varighet og legge til skala. De akustiske egenskapene i algoritmen ble derfor følgende:

- Dansbarhet(Danceability)
- Energi(Energy)
- Nøkkel(Key)
- Lydstyrke(Loudness)
- Tempo(Tempo)
- Taktart(Time Signature)
- Skala(Mode)

De syv akustiske egenskapene i algoritmen ga en bred oversikt over generelle musikalske-egenskaper i en sang, uten å gi unødvendig redundans eller kompleksitet.

Det neste steget innebar å implementere beregningene av disse egenskapene i algoritmen. Alle egenskapene utenom skala inneholdt verdier representert som desimaltall eller heltall. Den egenskapen som skilte seg ut var skala. Derfor måtte skala bli håndtert annerledes, ettersom dens verdi er boolsk(0 eller 1). Dette ble løst ved at to sanger måtte ha lik verdi for skala, 0 for moll eller 1 for durr, for at de i det hele tatt skulle antas å være like. Som nevnt tidligere økte dette

nøyaktigheten drastisk, ettersom sanger var nødt til å ha lik skala.

De 6 øvrige egenskapene sine verdier var annerledes enn skalaen. For å kunne sammenligne disse egenskapene, måtte det avgjøres hvor like verdiene måtte være for å avgjøre om de var like eller ikke. Målet var å sette en minimumsgrense på rundt 70-90 % likhet mellom egenskapene for å avgjøre om to sanger var relatert i forhold til en egenskap. Hvordan prosentandelen skulle måles var noe usikkert, derfor ble det utprøvd to metoder.

Den første metoden var å bruke standardavvik for å se hvor like to egenskapers verdi var. Standardavviket er et mål for spredningen av verdiene i et datasett. Derfor ble det først regnet ut standardavviket for alle egenskapenes verdier. Så ble egenskapens verdi for de to sangene som skal sammenlignes sett opp mot hverandre, og hvis de var innenfor standardavviket for egenskapen, så ville de være "like". Dette ble også for generelt og resulterte i "like" sanger som egentlig ikke var like. Derfor ble det samme utprøvd ved å redusere standardavviket til 1/10, som resulterte i et mer fokusert sett resultater som fortsatt var stort, men også nøyaktig.

Den andre metoden som ble testet er den vanligste metoden for prosentregning, hvor man sjekker hvor mange prosent x er av y ved formelen: $p = \frac{x}{y} \times 100$.
Ved første forsøk med prosent ble det forsøkt med at egenskapenes verdier skulle være 60% eller mer for å være like, altså $p >= 60$. Dette fungerte, men resulterte i at hver sang hadde i snitt 200 like sanger. Det er mulig at dette resultatet kunne blitt brukt videre, ettersom det er et datasett på 4, 500 sanger og det er en høy sannsynlighet for at mange sanger høres like ut. Spesielt hvis man antar at pop-sanger i en forstand er mer like enn sanger generelt.
Prosenten ble økt til 70%, slik at sammenligningene ville bli mer presise. Dette resulterte i en algoritme som leverte et sett resultater som fremdeles var stort, men mer nøyaktig.

Etter nøye vurdering viste det seg at prosentregning var enklere i bruk enn standardavvik og returnerte mer presise resultater. På grunn av bruksmåten til standardavvik resulterte det i at likheten mellom sanger var avhengig av selve datasettet, men det er ikke ønskelig i denne sammenhengen. Ettersom to sanger høres ikke mer like ut på bakgrunn av spredningen av egenskapenes verdier i datasettet. Sangers likhet bør derfor kun bli regnet ut ifra hvor like verdiene er.

For alle beregningene, ble det brukt en boolsk verdi for å avgjøre om hver beregning var lik eller

ikke. Det ble vurdert å bruke flere grader av likhet for hver beregning, men dette ble ikke implementert. Å ha mer enn en grad ville komplisert hvordan relevante sanger ville vises i en Force-directed graph, se Figur 4.6. Man ville vært nødt til å vektlegge kanter mellom sanger, noe som trolig ville ført til ytterligere forvirring for mange brukere.
Videre antas det at grader av likhet ikke ville fungert optimalt. På grunn av at to sanger med en veldig høy grad likhet for lydstyrke, men lav grad for taktart ville muligens vært like med denne metoden. Dette stemmer ikke overens med hvordan musikk høres ut, en "rask" sang vil ikke være lik en "treg" sang. I tillegg, ville det matematiske regnestykket for å oppnå en likhet med flere grader vært langt mer komplisert enn det som var nødvendig for kun en grad.

Den endelige algoritmen returnerte en boolsk verdi for om to sanger var like eller ikke. Hvis den var sann, vil det si at verdiene for alle de 6 egenskapene dansbarhet, energi, nøkkel, lydstyrke, tempo og taktart var minst 70% like for de to sangene og at de hadde lik skala, altså enten durr eller moll. Falsk ble derfor returnert hvis maksimum en av sammenligningene for de 7 egenskapene var falsk.

4.8 Termfrekvens – invers dokumentfrekvens

Vektormodellen med TF-IDF vekting ble implementert for å finne ut hvilke sanger i kolleksjonen som er relaterte/like basert på sangtekst. For å oppnå dette målet om likhet med TF-IDF ble det regnet ut TF-IDF vekter for alle sangtekstene sine termer i kolleksjonen. Videre ble sangtekstene representert som vektorer i et t-dimensjonalt rom, hvor t representerer antall unike termer i kolleksjonen av sanger. Vektorene av sangtekstene ble så parvis sammenlignet ved bruk av cosinuslikhets-funksjonen. Cosinuslikhets-funksjonen regnet ut cosinus mellom de to vektorene og returnerte en verdi som sier noe om forholdet mellom de to vektorene, altså hvor like de er.

Formelen for TF-IDF er gitt ved $tfidf_{i,j} = tf_{i,j} \times idf_i$, hvor $tf_{i,j}$ representerer termfrekvensen for termen i i dokument(sangtekst) j og idf_i representerer den inverse-dokumentfrekvensen til termen i. Ettersom TF-IDF er produktet av $tf_{i,j}$ og idf_i, måtte disse metodene implementeres først.

Termfrekvens

Implementasjon av termfrekvens ved formelen $tf_{i,j} = f_{i,j}$, skal telle antall forekomster av term i i dokument(sang) j.

```
public double tfCalculator(String termToCheck, String[] totalterms) {
	double count = 0;  //telle forekomster av termen termToCheck i totalterms
	for (String term : totalterms) {
		if (term.equalsIgnoreCase(termToCheck)) {
			count++;
		}
	}
	return count / totalterms.length; //lengdenormalisering
}
```

Kodesnutt 4.7: Implementasjon av Termfrekvens(TF)

I Kodesnutt 4.7 ovenfor tar metoden tfCalculator to parameter. Det første er termen som skal vektes med termfrekvens og det andre er et dokument(sang) som termen vil vektes ut ifra. Metoden teller antall forekomster av termen i dokumentet og returnerer en lengdenormalisert TF-vekt. Det er valgt å bruke lengdenormalisering, ettersom sangtekster kan være av forskjellig lengde. Uten en lengdenormalisert TF-vekt ville lengre dokumenter ha en fordel i gjenfinning over korte dokumenter, som diskutert i seksjon 2.1.1.

Invers dokumentfrekvens

Implementasjon av invers dokumentfrekvens ble gjort utifra formelen for idf:

$idf_i = \log \frac{N}{|\{d \in D : t_i \in d\}|}$, hvor N er antall dokumenter i samlingen og $|\{d \in D : t_i \in d\}|$ er antall dokumenter d i kolleksjonen D hvor termen t_i oppstår.

```
public double idfCalculator(String termToCheck, List<String[]> allDocuments) {
	double count = 0;//telle forekomster av termen termToCheck I allDocuments
	for (String[] document : allDocuments) {
		for (String term : document) {
			if (term.equalsIgnoreCase(termToCheck)) {
				count++;
				break;
			}
		}
	}
	return Math.log(allDocuments.size() / count);
}
```

Kodesnutt 4.8: Implementasjon av Inversdokumentfrekvens(IDF)

Java-implementasjonen for IDF-formelen er vist i Kodesnutt 4.8.
Metoden ”idfCalulator” tar to parameter, det første er termen som skal vektes med invers dokumentfrekvens og det andre er en liste av alle dokumeter(sanger) i kolleksjonen. Metoden teller antall dokumenter termen forekommer i og returnerer termen sin IDF-vekt, ved å ta logaritmen med base 2 av antall dokumenter(sanger) i kolleksjonen delt på antall dokumenter termen forekommer i.

Termfrekvens – invers dokumentfrekvens

Java-implementasjonen av TF-IDF vekten til en term er vekten returnert av tfCalculator × vekten returnert av idfCalculator for en gitt term. Følgende eksempel, Kodesnutt 4.9, genererer en representasjon av et vektor-rom bestående av alle sanger representert som vektorer, med tilhørende tf-idf vekting av termene som vektorens dimensjoner.

```
public void tfIdfCalculator() {
	double tf; //termfrekvens
	double idf; //inversdokumentfrekvens
	double tfidf; //tfifd
	for (String[] document : termsDocsArray) {
		double[] tfidfvector = new double[allTerms.size()];
		int count = 0;
		for (String term : allTerms) {
			tf = new TfIdf().tfCalculator(term, document);
			idf = new TfIdf().idfCalculator(term, termsDocsArray);
			tfidf = tf * idf;
			tfidfvector[count] = tfidf;
			count++;
		}
		tfidfDocsVector.add(tfidfvector);  //lagrer vektoren;
	}
}
```

Kodesnutt 4.9: Implementasjon av Termfrekvens-Invers dokumentfrekvens (TF-IDF)

Metoden ”tfIdfCalculator” i Kodesnutt 4.9 itererer gjennom alle dokumentene(sangene) i kolleksjonen termsDocsArray og genererer en vektor med t-dimensjoner, hvor t representerer antall unike termer i kolleksjonen. Så itererer metoden gjennom antall unike termer i kolleksjonen allTerms, for å kalkulere tf-idf vekten for termen i dokumentet og lagrer det i tilhørende dimensjon i vektoren. Når dette er utført for alle termer i kolleksjonen lagres den t-dimensjonale vektoren i listen tfidfDocsVector som representerer et vektorrom.

For å så kunne finne ut hvor like vektorene i vektor-rommet er, måtte cosinuslikhets-funksjonen implementeres.

Cosinuslikhets–funksjonen

Java-implementasjonen for å regne ut cosinus-likheten mellom to vektorer er basert på den matematiske formelen for cosinuslikhet: $sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|}$. Hvor $\vec{d_j} \cdot \vec{q}$ er det euklidske indreprodukt av de to vektorene, altså produktet av vektene $w_{i,j}$ og $w_{i,q}$ for alle vekter i de to vektorene. $|\vec{d_j}| \times |\vec{q}|$ er produktet av den euklidske norm til d_j og q. Java-implementasjonen vises i følgende kodesnutt, Kodesnutt 4.10.

```
public double cosineSimilarity(double[] docVector1, double[] docVector2) {
	double eucInnerProduct = 0.0;
	double eucNorm1 = 0.0;
	double eucNorm2 = 0.0;
	double cosineSimilarity = 0.0;
	for (int i = 0; i < docVector1.length; i++)
	{
		eucInnerProduct += docVector1[i] * docVector2[i]; //a.b
		eucNorm1 += Math.pow(docVector1[i], 2); //(a^2)
		eucNorm2 += Math.pow(docVector2[i], 2); //(b^2)
	}
	eucNorm1 = Math.sqrt(eucNorm1); //Kvadratrot(a^2)
	eucNorm2 = Math.sqrt(eucNorm2); //Kvadratrot(b^2)
	if (eucNorm1 != 0.0 | eucNorm2 != 0.0) {
		cosineSimilarity = eucInnerProduct / (eucNorm1 * eucNorm2);
	} else {
		return 0.0;
	}
	return cosineSimilarity;
}
```

Kodesnutt 4.10: Implementasjon av cosinuslikhets-funksjonen

Metoden cosineSimilarity i Kodesnutt 4.10 tar to parameter, to forskjellige vektorer, og returnerer cosinusverdien av vinkelen mellom de to vektorene. For å komme frem til cosinusverdien itererer metoden gjennom alle dimensjonene i de to vektorene for å regne ut produktet av deres euklidske indreprodukt og hver deres euklidske norm. Det er kun nødvendig å bruke en av de to vektorene sin størrelse for å iterere gjennom begge, ettersom vektorene vil uansett ha like mange dimensjoner, altså antall unike termer i kolleksjonen. Hvis ikke en av de to euklidske normene er 0.0, altså at en

av vektorene er 0-dimensjonale(noe som mest sannsynlig, aldri vil skje), vil cosinusverdien bli utregnet. Hvis tilfellet er at en vektor ikke har noen dimensjoner vil cosinusverdien være 0.0, altså ikke like.

4.9 Latent semantic indexing

Kort forklart omhandler latent semantic indexing om å redusere singulærverdi-dekomposisjon av termfrekvens matrisen t x d, X, til en k-dimensjonal matrise. Hvor $t \gg k$, t er antall termer og d er antall dokumenter. Å implementere dette i Java krever flere avanserte lineære algebraiske funksjoner. For å forenkle implementasjonen av disse har Jama biblioteket[9] sin Matrix(matrise) klasse blitt brukt.

Jama sin Matrix klasse gir de grunnleggende operasjonene til lineær algebra . Klassen har ulike konstruktører som gjør det enkelt å produsere Matrix matriser fra Java sine todimensjonale matriser(arrays) av dobbel presisjon flyttall(double). Matrix matrisene har forskjellige get- og set-metoder som enkelt gir tilgang til submatriser og matriseelementer.

De grunnleggende aritmetiske operasjonene Jama støtter er matrise addisjon og multiplikasjon , samt matrisenormer. Jama har også klassen SingularValueDecomposition hvor konstruktøren utfører en singulærverdi-dekomponering av en Matrix matrise. Å bruke Jama-biblioteket til å utføre "tunge løft" har gjort det enklere og raskere å implementere lineære LSI.

```
public Matrix transformToLSI(Matrix matrix) {
	SingularValueDecomposition svd = new SingularValueDecomposition(matrix);
	Matrix wordVector = svd.getU();
	Matrix sigma = svd.getS();
	Matrix documentVector = svd.getV();
	Matrix reducedWordVector = wordVector.getMatrix(
		0, wordVector.getRowDimension() - 1, 0, k - 1);
	Matrix reducedSigma = sigma.getMatrix(0, k - 1, 0, k - 1);
	Matrix reducedDocumentVector = documentVector.getMatrix(
		0, documentVector.getRowDimension() - 1, 0, k - 1);
	Matrix weights = reducedWordVector.times(
		reducedSigma).times(reducedDocumentVector.transpose());
	return weights;
}
```

Kodesnutt 4.11: Implementasjon av LSI ved redusert singulærverdi-dekomposisjon(SVD)

[9] http://math.nist.gov/javanumerics/jama/

Kodesnutt 4.11 viser LSI-implementasjonen i Java ved bruk av Jama-biblioteket. Metoden transformToLSI tar et parameter, dette er en termfrekvens matrisen som det ønskes å vekte som LSI. Først blir SVD-matrisen til termfrekvens matrisen generert av SingularValueDecomposition-konstruktøren. For å så kunne redusere SVD matrisen blir hvert av de tre komponentene/submatrisene i SVD-matrisen uthentet. Hvert av disse komponentene blir så redusert til en k-dimensjonal matrise, før den opprinnelige matrisen blir rekonstruert ved produktet av de tre reduserte k-dimensjonale matrisene. Metoden returnerer derfor den rekonstruerte matrisen av den reduserte k-dimensjonale SVD matrisen til den opprinnelige matrisen.

Den implementerte versjonen av LSI kan fungere på dokumenter fra hvilken som helst kolleksjon. Ettersom metoden byger på lineære algebraiske funksjoner til en gitt matrise, kan operasjonene utføres på en hvilken som helst rektangulær matrise. Det er derfor lett å gjenbruke LSI-metoden om det i fremtiden vil være interessant å finne artister eller år som ligner på hverandre, basert på lyriske data. Man er kun nødt til å produsere matrisen som det skal kalkuleres LSI for, som raskt kan gjøres ved hjelp av Jama klassen Matrix sin konstruktør.

4.10 Datavisualisering

Datavisualisering blir brukt for å utnytte menneskets intuitive evne til å gjenkjenne struktur og mønstre. For å realisere dette er det viktig at visualiseringene er unike, slik at de kun kommuniserer de viktigste punktene. Det vil si at predefinerte visualiseringer, som Google Motion Charts og Tableau, ikke vil benyttes. Implementasjon av visualiseringene i dette prosjektet benytter seg derfor av JavaScript bibliotekene D3.js[10] og Highcharts[11]. Disse bibliotekene gjør det mulig å oppfylle det ikke-funksjonelle kravet om tilgjengelighet. Det vil si at visualiseringene er tilgjengelig i en hver moderne nettleser og krever ikke noen form for tilleggsverktøy.

D3(Data-Driven Documents) også referert til som d3.js er et åpent Javascript bibliotek som brukes for å lage datavisualiseringer på Internett. D3 bringer data til liv ved hjelp av web-standarder som HTML, SVG, og CSS. D3 sin vektlegging på webstandarder gir alle muligheter i moderne nettlesere uten å binde seg til et proprietært rammeverk, og kombinerer kraftige visuelle komponenter og en data-drevet tilnærming til DOM manipulasjon. D3-biblioteket er perfekt for å lage helt unike visualiseringer, der man er nødt til å skrive kode for å generere hver eneste strek i en

[10] http://d3js.org/
[11] http://www.highcharts.com/

graf. Det er derfor blitt brukt til å implementere de mest unike visualiseringene, slik som force directed graph og ordsky, se seksjon 4.10.2 og 4.10.1.

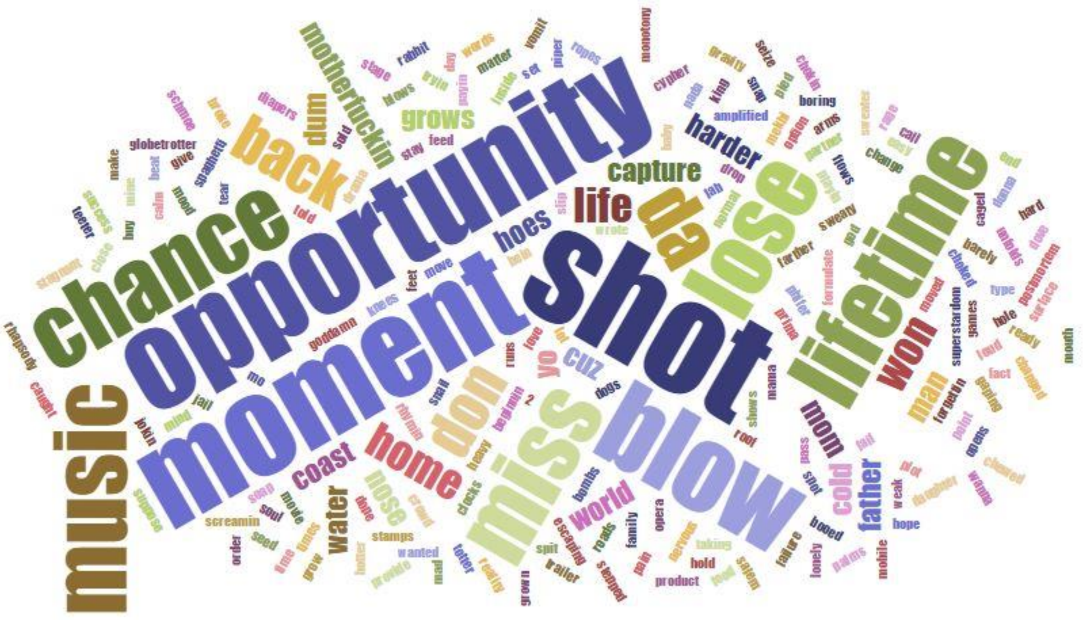
Highcharts derimot kommer med flere pre-definerte elementer som enkelt kan modifiseres for å kommunisere det man selv ønsker. Highcharts er et visualiserings bibliotek skrevet i ren Javascript, og tilbyr en enkel måte å legge til interaktive diagrammer til en nettside eller webapplikasjon. Highcharts støtter flere typer diagrammer, blant annet, boble, linje, stolpe, sektor og venn diagramtyper. Det er gratis å bruke for ikke-kommersielle aktører, slik som akademikere og for non-profit organisasjoner.

Det å lage gode og interaktive datavisualiseringer har vært krevende, ettersom det ikke finnes noen "best-practice" metode for å presentere et gitt datasett. Den beste måten å presentere et unikt datasett på er avhengig av selve datasettet og hva man ønsker å formidle med visualiseringen. Hvert datasett har derfor blitt visualisert på flere forskjellige måter, før det har blitt funnet en visualisering som passer godt og kommuniserer det som er ønskelig. På bakgrunn av denne utviklingsprosessen har det gått hånd i hånd med iterativ utviklingsmetodikk med korte iterasjoner, «release early, release often». Dette har gjort det mulig å vise arbeidet til andre, få reaksjoner, se om det genererer den type respons som antatt, også iterere.

Videre i denne seksjonen vil de visuelle komponentene brukt i systemet presenteres. Det vil vises hvordan de ser ut og diskuteres hvordan de bidrar til å forenkle presentasjonen av data.

4.10.1 Ordsky

For å visualisere sangtekster er det gjort et forsøk på å utvikle en såkalt ordsky(Word Cloud). En ordsky er en spesiell visualisering av tekst, der de mest brukte ordene er fremhevet i representasjonen. Visualiseringen avslører frekvensene av de ulike ordene som vises i et stykke tekst. Det er en spennende presentasjon av tekstlig data og gir brukeren en forståelse av den generelle sammensetning av de ofte brukte ordene. Dette lar seerne til en viss grad ha en oversikt over de viktigste emnene og de viktigste temaene i en tekst, og kan illustrere de viktigste standpunktene holdt av forfatteren av teksten.



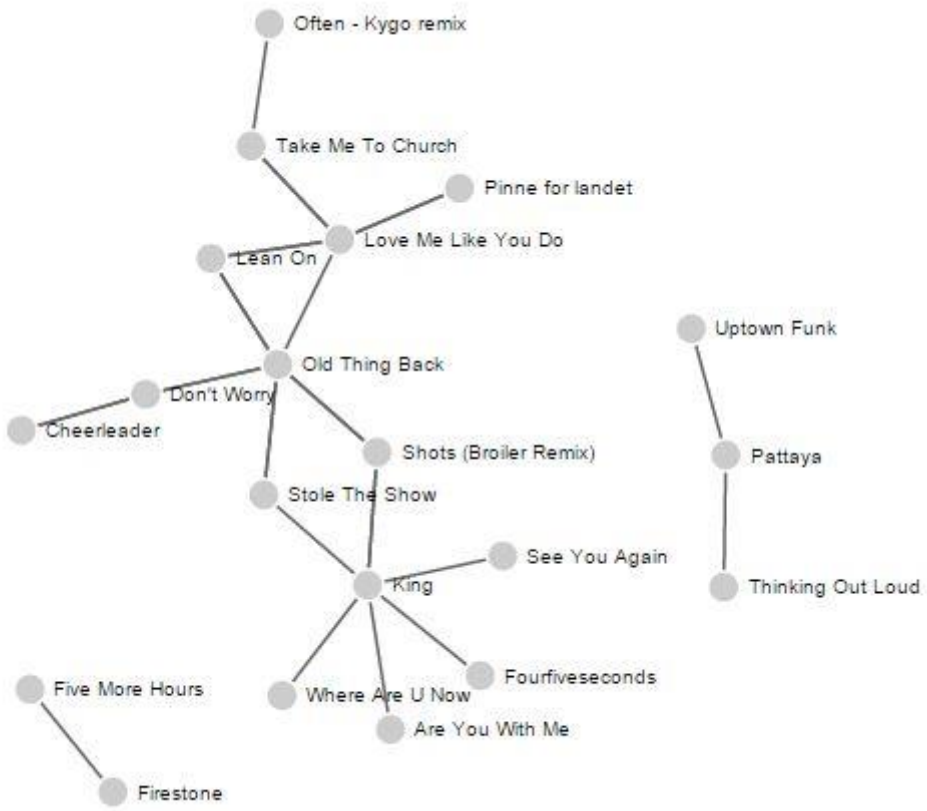
Figur 4.5: Implementasjon av ordsky for sangen Lose Yourself av Eminem

Figur 4.5 er en implementasjon av ordsky-visualiseringen for sangen Lose Yourself av Eminem. Ordskyen er ikke ulik den man ser på flere nettsteder, hvor populære emner blir vist i en liste skalert etter ordfrekvens. Man får et interessant innblikk i sangens tekst og muligens mening. En negativ ting ved ordskyen er at ordene er hentet ut av sammenhengen. Den hyppige omtalen av ord som "chance" og "opportunity" i Figur 4.5 er utilstrekkelig informasjon for å vite meningen bak disse ordene. Ordskyen kan derfor være misvisende fordi den neglisjerer semantikk av ordene og setninger ordene består av. Dette fører til at ordskyen mislykkes i å behandle en setning fra sangen, "This opportunity comes once in a lifetime yo", som et meningsfylt uttrykk i seg selv. Det er derfor ikke anbefalt å bruke ordskyer som verktøy for en semantisk sammenligning av tekstlig data, men den er perfekt for å gi brukere en rask introduksjon til teksten med sin kvantitative oversikt over brukte ord.

4.10.2 Force-directed grapf

For å støtte en visualisering av like sanger ble det implementert en visualisering ved å bruke Force-directed graph. Dette er en type visualisering som appellerer til forskere fra mange forskjellige felt og står for dusinvis av tilgjengelige implementasjoner. Oppsettet til en slik graf blir beregnet ut ifra en type av algoritmer, kalt force-directed algoritmer. Helt siden 1963 har det blitt forsket på slike

algoritmer og nye varianter er fortsatt introdusert hvert år. Grafer tegnet med disse algoritmene pleier å være estetisk tiltalende og eksponere symmetri (Kobourov, 2012, s. 1).
I Figur 4.6 vises et eksempel av den implementerte force-directed grafen som blir brukt i applikasjonen for å se relasjoner mellom sanger.



Figur 4.6: Implementasjon av force-directed graf

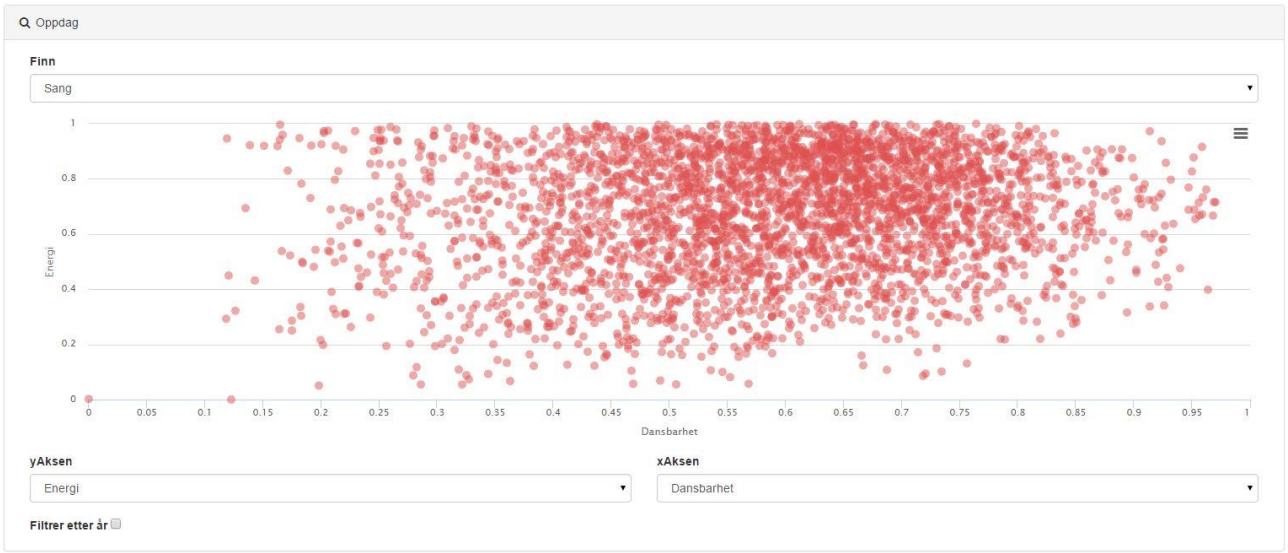
Figur 4.6 er generert på bakgrunn av et testsett for å se hvordan representasjonen ville bli seendes ut. Visualiseringen er estetisk tiltalende og gir seeren en oversiktlig presentasjon av de forskjellige sangene og deres relasjoner. Grafen er urettet, noe som passer perfekt med sangers relasjoner i systemet, ettersom relasjonene er gjensidige. Altså hvis sang A har en relasjon til Sang B er også Sang B nødt til å være relatert A. Kantene mellom nodene er alle vektet likt, dette på grunnlag av forenklingen gjort i den implementerte sammenligningsalgoritmen presentert i seksjon 4.7, hvor det ble valgt å representere en relasjon som en boolsk verdi.

I Figur 4.6 er tittelen til sangen plassert ved siden av noden, slik at man raskt kan identifisere de forskjellige sangene. Det ble lekt med med tanken på å bruke denne typen visualisering for å vise alle 4,500 sangene slik at man kunne få en overordnet oversikt. Resultatet ble dessverre for vanskelig å lese når veldig mange noder ble gruppert sammen og titler la seg oppe på hverandre.

Det var også en veldig lang ventetid for å laste inn 4, 500 sanger, noe som førte til at ideen ble forkastet.

4.10.3 Spredningsplott

For å utvikle en visualisering som skulle la brukeren oppdage trender, klynger, mønstre og sammenhenger mellom objekter i datasettet ble det gjort et forsøk på å presentere det i et spredningsplott. Spredningsplottet visualiserer korrelasjonen mellom to variabler x og y, for eksempel dansbarhet og tempo. De individuelle datapunktene er representert i et todimensjonalt rom, hvor aksene representerer variabler (x på den horisontale aksen og y på den vertikale aksen). Spredningsplottet fungerer godt til å vise trender, klynger, mønstre og sammenhenger i en sky av datapunkter, spesielt i store datasett, se Figur 4.7



Figur 4.7: Implementasjon av spredningsplott.

Spredningsplottet støtter oppdagelse av nye objekter. Kun ved å endre på de forskjellige nedtrekkslistene muliggjør det å finne objekter og se mønstre mellom de. I det implementerte spredningsplottet i Figur 4.7 kan man enkelt endre hvilke objekter man ønsker å oppdage, ved å velge sang, artist eller liste i den øverste nedtrekkslisten, kalt "Finn". Ved å bruke de øvrige nedtrekkslistene kan man endre x og y aksen. Ved å klikke på "Filtrer etter år"-avkrysningsboksen som er plassert nederst i det venstre hjørnet vil det komme frem en glidebryter(slider) som gjør det mulig for brukeren å filtrere spredningsplottet etter år, fra 1960-2014. Nedtrekkslistene og

avkrysningsboksen gir brukeren flere tilnærminger å se datasettet på. Brukeren kan for eksempel sortere etter tempo og varighet, for å så filtrere etter år. Det tillater brukerne å manipulere datasettet til meningsfulle visualiseringer, noe som åpner for at brukeren selv kan velge hvilke egenskaper ved objektene han ønsker å se på.

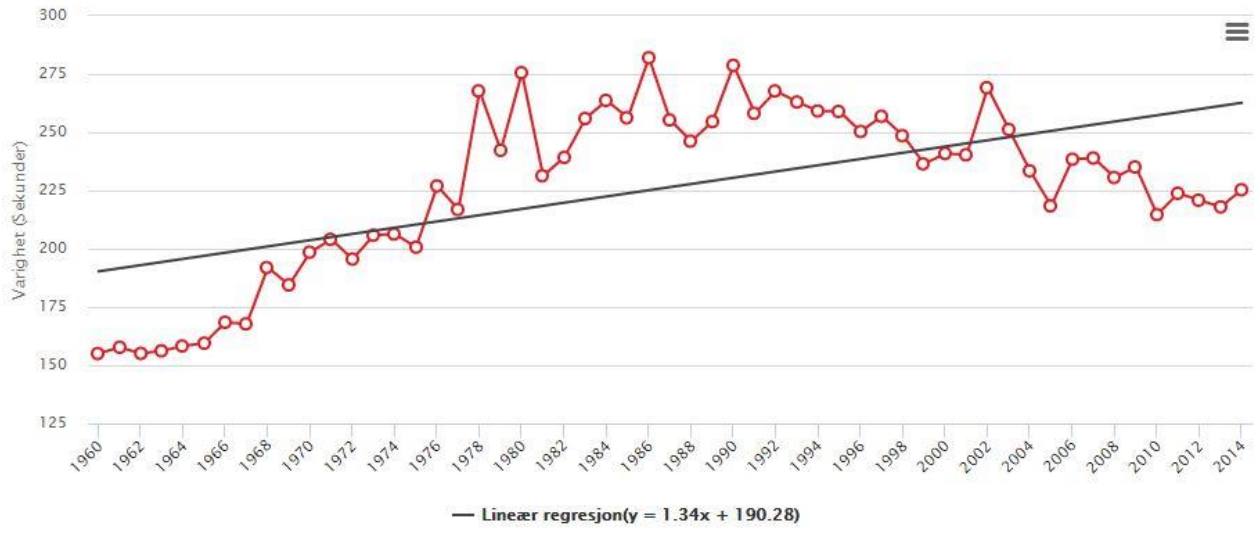
Hvis brukeren drar musen over et av objektene i spredingsplottet vil han også få presentert data om objektet, x og y egenskapenes verdier og objektets navn, for eksempel "Artist: Madonna, Tittel: Like a virgin, Dansbarhet:0.5, Energi:0.2".

4.10.4 Andre Visualiserings typer

Linjediagram

For å vise endringer i data over tid er det blitt implementert et linjediagram. Linjediagrammet sammenligner to variabler i et koordinatsystem. Hvor hver variabel blir plottet langs en av de to aksene, x(horisontal) eller y-aksen(vertikal). Hvert punkt tilsvarer en observasjon på et gitt tidspunkt. For å fremheve endringene i dataene, trekkes det linjer mellom disse punktene.

Linjediagrammet har vist seg å være spesielt egnet for å vise trender i data på en oversiktlig og tydelig måte. Det viser synlig hvordan en variabel påvirkes av den andre, ut ifra om den øker eller minsker. Diagrammet gjør det også mulig for brukere å komme opp med antakelser om resultater av data som ennå ikke er registrert. Dette illustreres i Figur 4.8.



Figur 4.8: Implementasjon av linjediagram fra år 1960 til 2014

Figur 4.8 presenterer den gjennomsnittlige verdien av sangers varighet fra år 1960 og til 2014. Som nevnt tidligere gjør karakteristikkene til linjediagrammet det mulig å enkelt konkludere med at sanger har blitt lengre opp gjennom årene. Det er også mulig å komme opp med egne antakelser om dataene, slik som at sangers varighet ser ut til å være på sitt lengste i 1986 og at det vil komme til å fortsette å synke i fremtiden.
Det implementerte linjediagrammet i web-applikasjonen er interaktivt, slik at man kan bevege musen over et gitt punkt for å få mer informasjon. Man kan også endre akustiske egenskaper ifra en nedtrekksmeny, se Figur 4.11 .

Linjediagrammet i Figur 4.8 inneholder også lineær regresjonen til dataene presentert. Dette er den best plasserte rette linjen mellom punktene. I Figur 4.8 viser den prognosen(en forutsigelse om hvordan utvikling vil arte seg) til sanger sin varighet gjennom årene.

Kulegraf.

Som presentert ovenfor gir linjediagram en veldig enkel oversikt over større datasett, men for å se på sammenhengen mellom kun to verdier viste det å ikke være en god løsning. Et linje eller stolpediagram med kun to punkter tar opp mye plass og brukere forventer ofte mer enn to datapunkter i slike diagram. For å presentere mindre datasett er det derfor implementert kulegraf. Kulegrafen(Bullet Chart) sitt lineære og enkle design gir en rik presentasjon av data på en liten plass. Som de fleste metre og målere, måler kulegrafen et enkelt kvantitativt mål sammen med komplementære mål for å berike betydningen for det utvalgte målet. Et eksempel på bruken til en kulegraf er å presentere høyden til en person sammen med gjennomsnittshøyden for å gi mer mening til personen sin høyde. Sitt lineære design gjør at den passer godt inn på en nettside, ettersom den krever liten plass, samtidig som den er enkel å lese. Figur 4.9 viser dette i praksis.

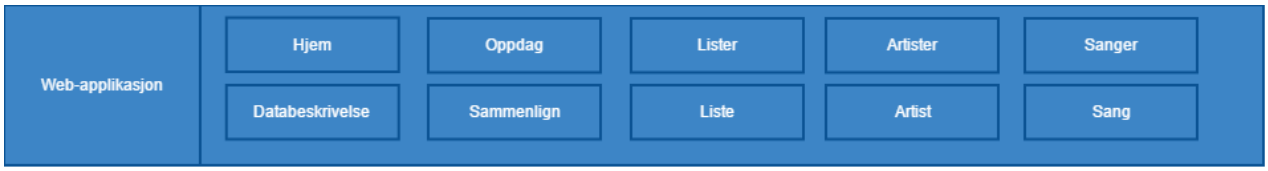
Figur 4.9: Implementasjon av kulegraf for den akustiske egenskapen dansbarhet

Kulegrafen representert i Figur 4.9 inneholder først og fremst en tekst-etikett for å identifisere hva

som måles i grafen. I eksempelet er det en sang sin akustiske egenskap "dansbarhet" som måles. Den sorte stolpen representerer primær dataen i grafen og er derfor visuelt fremtredende. Den grå stolpen er et komplementært mål som representerer gjennomsnittet av dansbarhet i alle sanger. Stolpen er mindre visuelt dominerende enn primær målet, ettersom den er mindre viktig. Begrunnelsen for å ha implementert denne grafen er at brukeren ikke nødvendigvis vet hva dansbarhetens verdi sier hvis den er presentert alene. Presentert sammen med gjennomsnittsverdien vil brukeren ha en bedre forståelse av verdien, om den er høy eller lav. I selve Web-applikasjonen er det mulig å dra musen over elementene for å få en beskrivelse av hva de forskjellige stolpene betyr og selve verdien, slik som "Gjennomsnitt: 0.4".

4.11 Web-applikasjonen

For å visuelt kunne presentere all innsamlet data er det implementert en front-end i systemet, i form av en web-applikasjon. Web-applikasjonen er utviklet med AngularJS[12], noe som gjør det enkelt å deklarere dynamiske visninger i applikasjonen. Web-applikasjonen består av flere nettsider, slik at det er mulig å fordele presentasjonen av dataene på en logisk måte. De nettsidene applikasjonen består av er illustrert i Figur 4.10 og listet i seksjon 4.2.6.

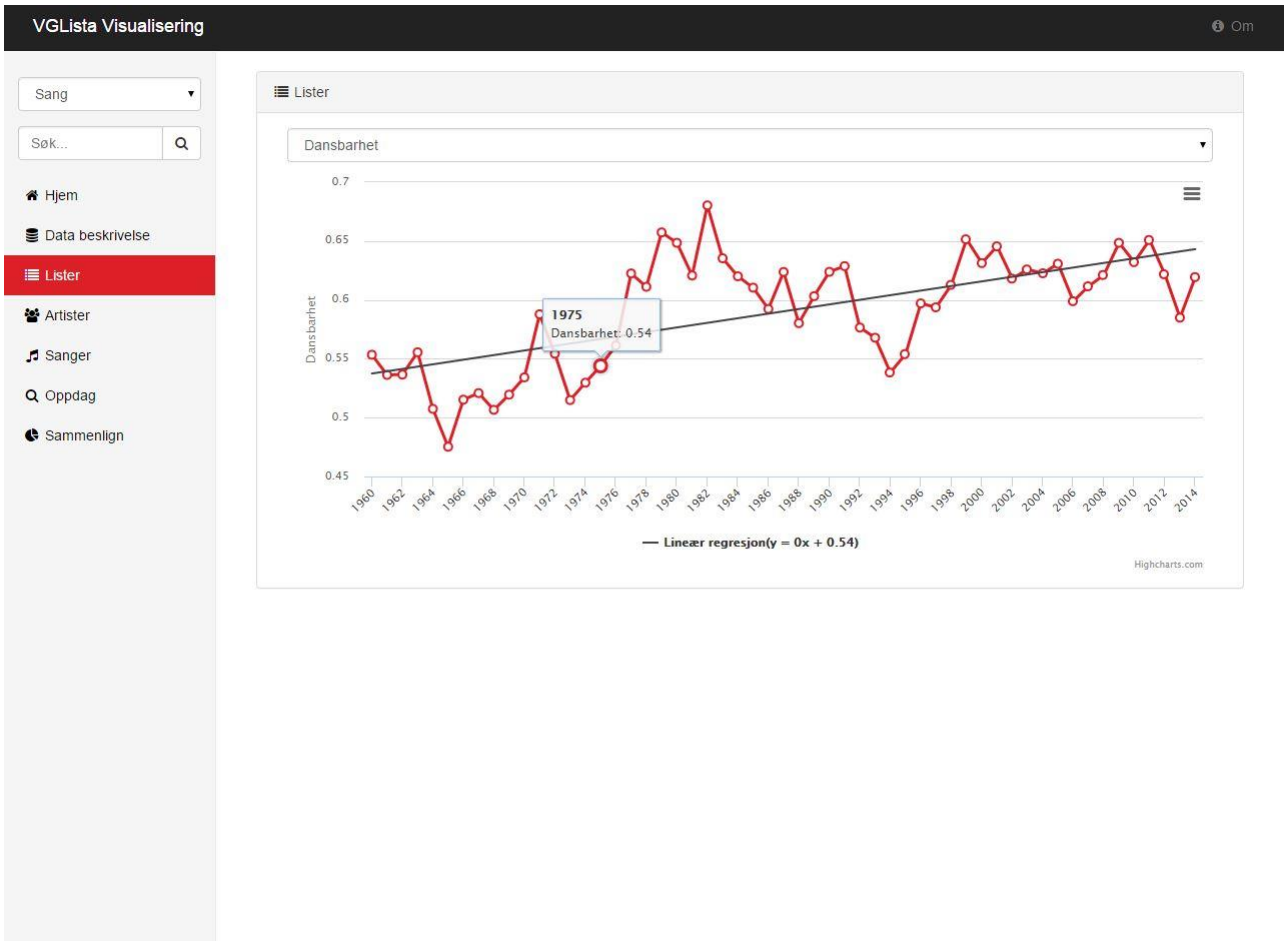


Figur 4.10: HTML-sidene i Web-applikasjonen

Gjennom utviklingen av sidene har det vært fokus på å skape en intuitiv og konsistent applikasjonen. Dette er realisert ved å bruke like metoder og visualiseringer på de forskjellige sidene. Slik at når brukere først har forstått en side sin struktur, vil de også kunne forstå de andre sidene. Videre i denne seksjonen vil de mest essensielle nettsidene presenteres. For presentasjon av de andre nettsidene, se appendiks A-F.

[12] https://angularjs.org/

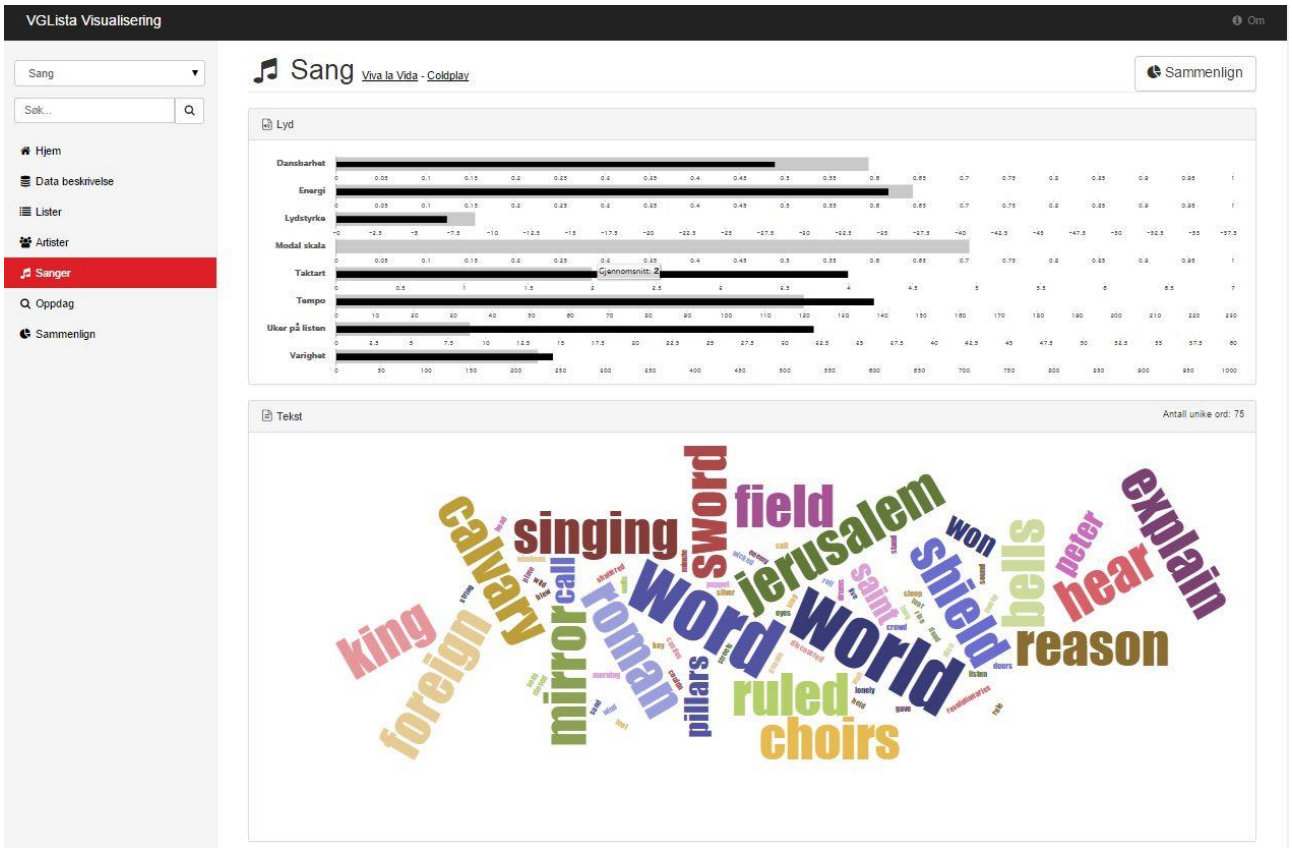
4.11.1 Lister



Figur 4.11: Skjermbilde av lister-siden i web-applikasjonen

Figur 4.11 er et skjermbilde av den implementerte lister-siden.
Siden presenterer endringer i akustiske egenskaper over årene i norsk popmusikk. Data er presentert i et linjediagram hvor det er mulig å endre verdier i y-aksen ved å endre egenskap i nedtrekkslisten, lokalisert over diagrammet. Nedtrekkslisten består hovedsakelig av akustiske egenskaper som dansbarhet, energi, osv. Det er også lagt til en ekstra egenskap kalt ”Liste Varighet”, som forteller hvor mange uker i gjennomsnitt en sang har vært på listen.
Linjediagrammet presenterer interessante trender i akustiske egenskaper over årene på en oversiktlig og tydelig måte. Utifra diagrammet kan brukere besvare spørsmål om norskpopmusikk og komme opp med antakelser om resultater av data som ennå ikke er registrert, basert på tidligere data.

4.11.2 Sang

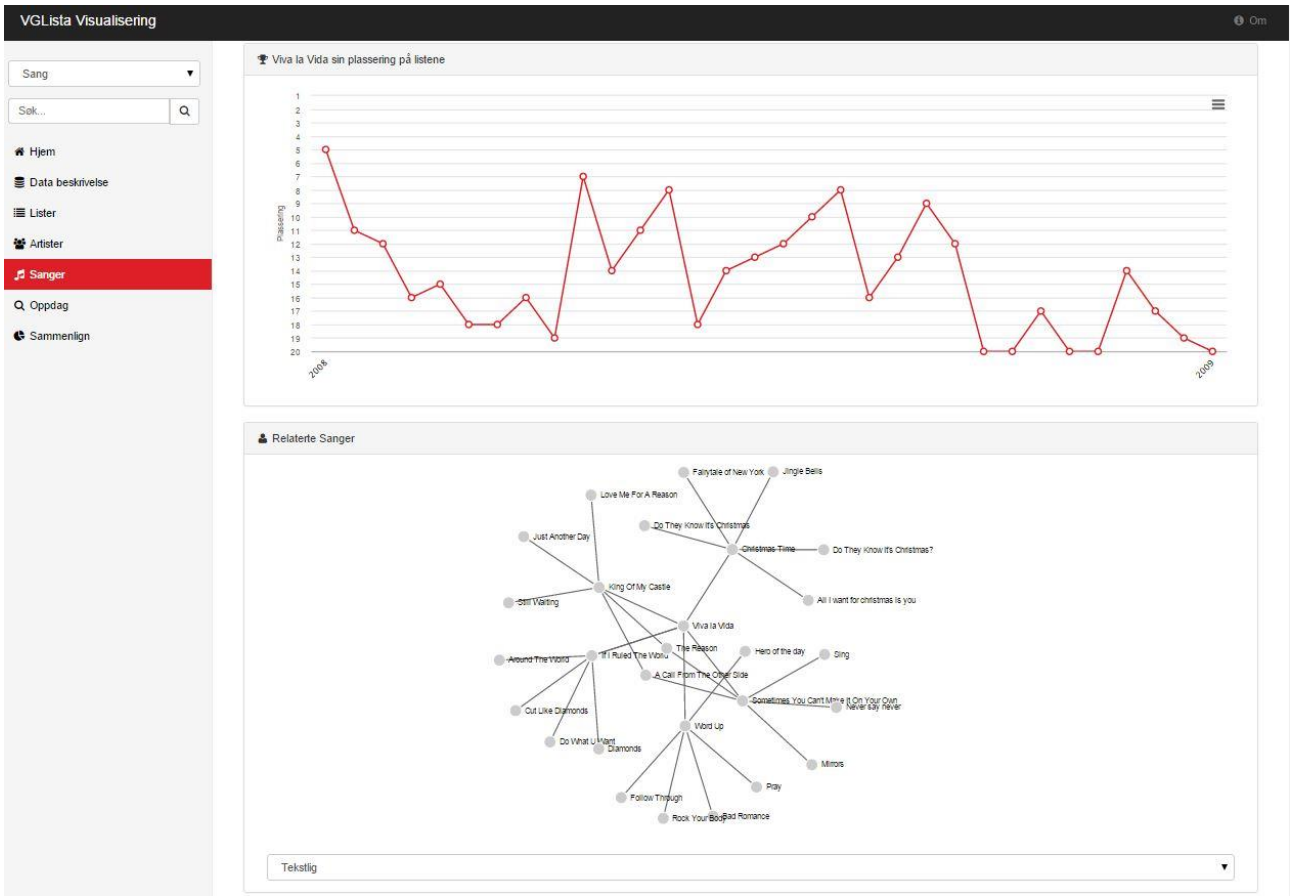


Figur 4.12: Skjermbilde av sang-siden, 1 av 2.

Sang-siden (Figur 4.12) presenterer data om en gitt sang. Sangens navn og artist er presentert i toppen av siden og fungerer begge som lenker. For å presentere de akustiske egenskapene slik at de kan sees i forhold til gjennomsnittsverdien i norsk popmusikk, blir kulegrafen brukt. Data om antall uker på listen en sang har vært, blir også presentert i en kulegraf. Kulegrafen er, som nevnt i seksjon 4.10.4, godt egnet til å berike betydningen for et utvalgt mål.

Presentasjon av sangteksten er implementert ved å bruke en ordsky, som er presentert i seksjon 4.10.1.Visualiseringen gir en spennende presentasjon av sangteksten og gir brukeren en forståelse av den generelle sammensetning av de ofte brukte ordene.

Ved å bevege seg videre nedover på siden blir brukeren presentert med et linjediagram, se Figur 4.13.



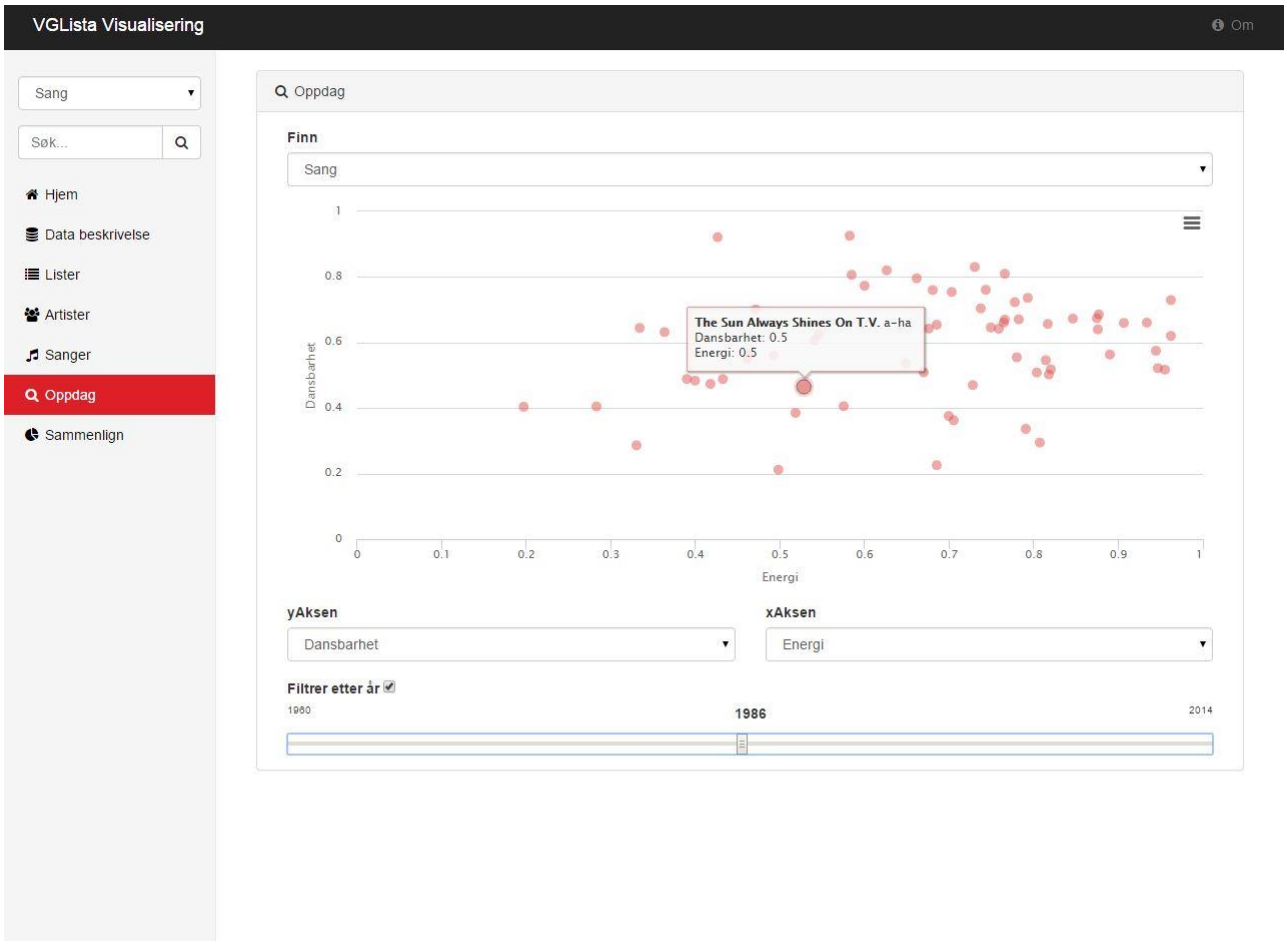
Figur 4.13: Skjermbilde av sang-siden, 2 av 2.

Linjediagrammet presenterer plasseringene til sangen opp gjennom årene på VG-Lista Topp 20 listene. Ved å bevege musen over de forskjellige datapunktene i diagrammet kommer det opp et lite vindu med informasjon om datapunktet, slik som “2008, Uke: 22, Plass: 12”.

Siden presenterer også de relaterte/like sangene til en gitt sang. Dette er implementert med en Force-directed graph, slik som vist i seksjon 4.10.2. For at brukeren skal ha mulighet til å endre mellom å se like sanger basert på tekstlig innholdt eller akustiske egenskaper er det implementert en nedtrekksliste hvor brukeren kan velge mellom enten tekstlig eller akustisk innhold.

Videre inneholder nettsiden en liste over andre sanger av artisten som har laget sangen, slik at brukere kan få en oversikt over sangene og navigere seg videre til andre sanger.

4.11.3 Oppdag

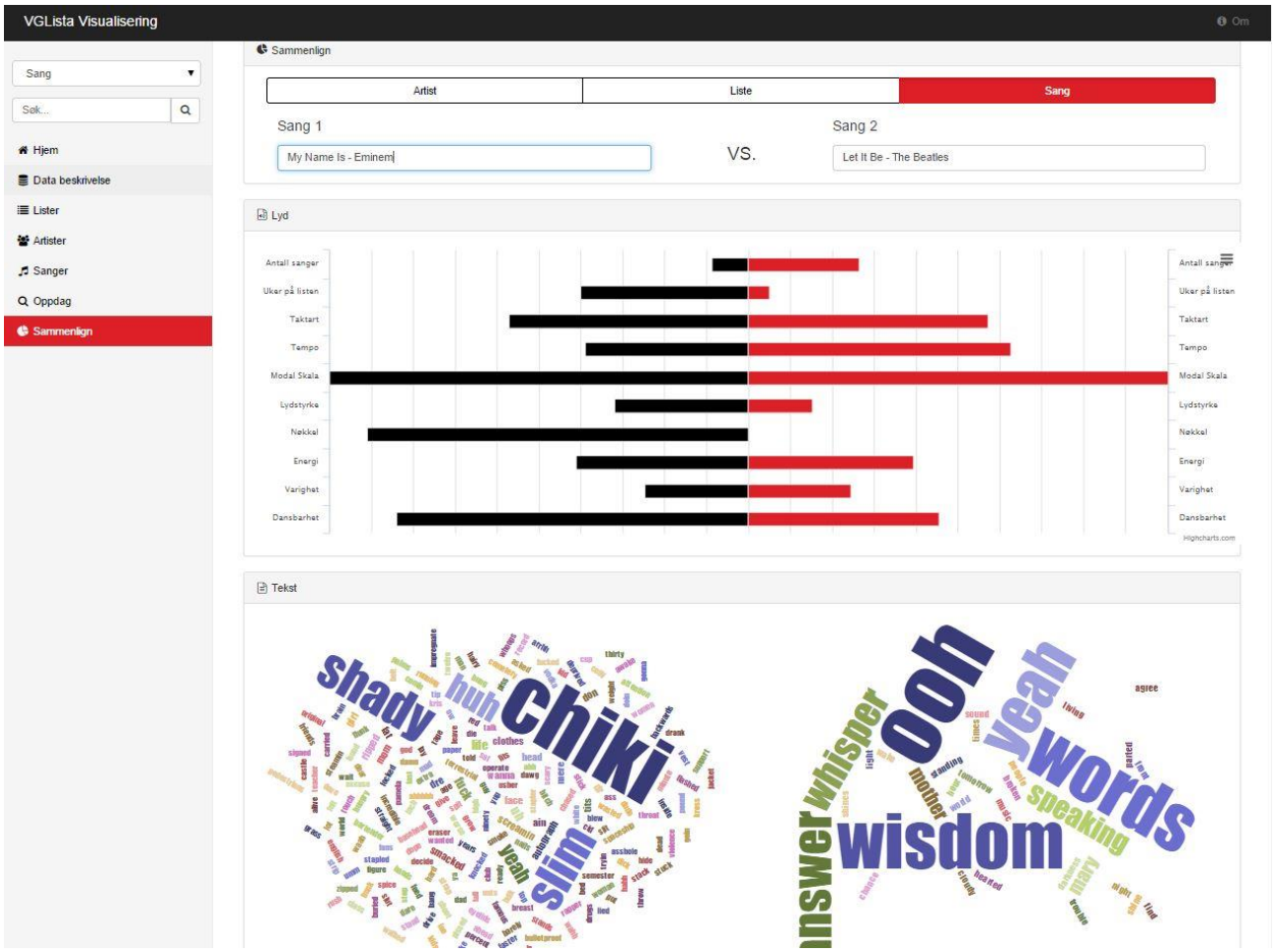


Figur 4.14: Skjermbilde av oppdag-siden

Siden inneholder et spredningsplott, som presentert i seksjon 4.10.4, for å la brukeren oppdage trender, klynger, mønstre og sammenhenger mellom objekter i norsk popmusikk.
Oppdagelse av forskjellige type objekter i spredningsplottet gjøres ved å velge fra nedtrekkslisten under ”Finn”-etiketten. Nedtrekkslisten sine valg er artist, liste eller sang.
Videre er det mulig å velge hvilke egenskaper man ønsker å sortere objektene etter, ved å endre verdiene for datapunktene sine x og y akser. Egenskapene man kan velge i nedtrekkslistene “yAksen” og “xAksen” er hovedsakelig akustiske egenskaper. Når brukeren velger egenskaper fra listene blir spredningsplottet automatisk oppdatert.
Et problem som oppstod i spredningsplottet var at det ofte ble uoversiktlig, grunnet mange datapunkt(objekter). Hvis man for eksempel oppdager sanger vil det presenteres ca 3,500 sanger som har akustiske egenskaper. For å gjøre det mer oversiktlig ble det implementert en form for filtrering av objekter, slik at spredningsplottet blir mer presist. Det er derfor gjort det mulig å filtrere

resultatene etter år, ved å klikke på avkrysningsboksen "Filtrer etter år". Et klikk på avkrysningsboksen fører til at det dukker opp en glidebryter(slider), som gjør det mulig å filtrere etter år. Hvis brukeren for eksempel endrer verdi i "slideren" gradvis, vil spredningsplottet filtreres og brukeren vil bli presentert med noe som kan minne om en animasjon.

4.11.4 Sammenlign



Figur 4.15: Skjermbilde av sammenlign-siden

Sammenlign-siden, vist i Figur 4.15, inneholder en funksjon for å sammenligne to objekter. Brukeren velger først hvilket type objekt han ønsker å sammenligne. Dette kan være artister, sanger eller lister. Valget gjøres ved å trykke på en av de tre knappene lokalisert øverst på siden. Videre velger for eksempel brukeren de to sangene han ønsker å sammenligne ved å skrive inn sangtitler for de to sangene i søkeboksene.

Resultatet presentert er to visualiseringer. Et stolpediagram med de forskjellige akustiske

egenskapene sine verdier presentert i forhold til hverandre. Tanken bak stolpediagrammet er at brukeren raskt skal få en oversikt over de akustiske egenskapene til de to sangene. Hvis brukeren ikke ser forskjell på størrelsen på stolpene, kan han bevege musen over stolpen for å få en mer presis beskrivelse av verdiene. Ordskyen er implementert slik som beskrevet i seksjon 4.10.1 og gir brukeren en oversikt over de mest brukte ordene i de to forskjellige objektene som sammenlignes.

4.12 Verktøy

Å realisere implementasjonen av systemet involverte bruk av flere forskjellige teknologier og verktøy. Det ferdige systemet har egen back-end og front-end. Dette innebærer oppgaver som å lagre og prosessere data, kommunisere med eksterne APIer, utføre avanserte beregninger og visuelt presentere data til brukeren. For at dette skulle fungere noenlunde smertefritt var det nødvendig å velge ut en god del verktøy og strukturere prosjektet godt, noe som var en tidskrevende oppgave. Heldigvis finnes det flere gode rammeverk som inneholder alle nødvendige verktøy, slik som MEAN-stacken.

4.12.1 MEAN-stack rammeverket

Mean[13] er et rammeverk for et enkelt utgangspunkt med MongoDB, Node.js, Express, og AnuglarJS baserte applikasjoner. Det er en fullstack JavaScript plattform for moderne webapplikasjoner og er rettet for alle slags JavaScript utviklere, både server og klient side. MEAN blander sammen de mest brukte og verdsatte teknologiene for JavaScript utvikling og legger fundamentet for å enkelt bygge komplekse web-applikasjoner. Dette har gjort det enkelt å presentere data fra back-end i front-end, ettersom begge deler benytter seg av Javascript-objekter, kalt JSON-objekter.

Alternativer til MEAN-stacken er den klassiske LAMP-stacken som inneholder komponentene Linux, Apache, MySQL, PHP. Selv om LAMP er den eldste stacken av de to, og er blitt vel-testet opp igjenom årene, falt valget på MEAN. Noen fordeler med å velge MEAN over LAMP er at det kun bruker et språk fra topp til bunn, nemlig Javascript og ettersom jeg har tidligere erfaringer med Javascript, både i front-end og back-end, forenklet dette utviklingsprosessen stort.

MEAN-stacken består av følgende komponenter:

[13] http://mean.io/#!/

MongoDB[14] er et skjemaløst NoSQL databasesystem. MongoDB lagrer data i binære JSON format. Dette gjør det enklere å sende data mellom klient og server, ettersom både klientside og serverside benytter seg av Javascript og JSON. Det har spart prosjektet for mye tid, ettersom det ikke har vært behov for å transformere data fra databasen til et annet format for å kunne presentere det.

Express[15] er et lettvekts rammeverk brukt til å bygge webapplikasjoner i NodeJS. Det tilbyr flere robuste funksjoner for å bygge single og multi side webapplikasjoner, noe som gjør det til en god løsning for Single-page-application, nettsider, hybrider, eller offentlige HTTP APIer. Express er blitt brukt i systemet for å muliggjøre kommunikasjon mellom klientsiden og databasen, slik at brukere av web-applikasjonen kan få tilgang til de data de ønsker.

AngularJS[16] er et åpen kildekode JavaScript rammeverk utviklet av Google som tilbyr rask og responsiv front end utvikling. Rammeverket søker å løse mange av de utfordringene som oppstår i utvikling av en Single-Page-Application(SPA). AngularJS er blitt brukt for å implementere selve Web-applikasjonen, den tar seg av presentasjon av data i dynamiske nettsider, kommunikasjon med Express APIet og brukeren sin interaksjon med Web-applikasjonen. AngularJS har også gått hånd i hånd med de andre JavaScript bibliotekene som er brukt til å visualisere data, slik som D3.js og Highchart.

Node.js[17] er en server side JavaScript plattform for skalerbare server-side og nettverks applikasjoner som gjør det mulig å kjøre applikasjoner skrevet i JavaScript. Node.js har derfor blitt brukt til å kjøre Express og MongoDB applikasjonene som er nevnt ovenfor. En stor fordel ved å ha brukt Node.js er at det inneholder et innebygd bibliotek for å tillate applikasjoner til å fungere som en webserver uten tilleggsprogramvare, slik som Apache HTTP Server.

Illustrasjonen i Figur 4.16 viser Mean-stacken sin struktur og det er enkelt å dra sammenligninger med den og systemetarkitekturen i Figur 4.2.

[14] https://www.mongodb.org/
[15] http://expressjs.com/
[16] https://angularjs.org/
[17] https://nodejs.org/

Figur 4.16: Komponenter av Mean-stacken, fra Silva (2014)

4.12.2 Bootstrap

For å raskt kunne utvikle visuelt estetiske nettsider i web-applikasjonen er det blitt brukt HTML, CSS og JavaScript rammeverket Bootstrap.

Bootstrap[18] er det mest populære HTML, CSS og JavaScript rammeverket for utvikling av responsive nettsider og gjør front-end webutvikling raskere og enklere, ettersom det inneholder flere predefinerte elementer for knapper, lister, osv. Ved å bruke Bootstrap og elementene som tilbys har det gått raskt å komme i gang med front-end utviklingen, og gjort det mulig å prøve ut flere designmønstre gjennom utviklingen av web-applikasjonen.

En av hovedfordelene ved å ha benyttet Bootstrap som rammeverk, er at det støtter utvikling av «mobile-first» nettsider og responsivt design. Det vil si at det har vært enkelt å gjøre web-applikasjonen delvis tilgjengelig for både mobiltelefoner og stasjonære datamaskiner. Bootstrap gjør dette mulig ved å bruke et dynamisk rutenett system som skalerer opp til 12 kolonner og justerer seg etter skjerm-størrelsen til enheten som besøker nettsiden. Det har gjort det enklere å oppnå det ikke-funksjonelle kravet om tilgjengelighet.

[18] http://getbootstrap.com/

5 Evaluering

Kapittelet begynner med å presentere en kvantitativ evaluering av gjenfinningsmetodene TF-IDF og LSI, hvor det finnes ut hvilken metode som presterte best i systemet. Videre er det gjennomført en usability-test for å vurdere bruken av systemet sin web-applikasjon. De kvalitative og kvantiative data generert fra usability-testingen er analysert for å svare på om web-applikasjonen er brukervennlig.

5.1 Evaluering av gjenfinningsmetodene

Evalueringen av gjenfinningsmetodene ble utført for å finne ut om den leksikalske tilnærmingen til TF-IDF eller den semantiske tilnærmingen til LSI var best egnet for å representere relasjoner mellom sangtekster i systemet.

Selve evalueringen har foregått ved å systematisk assosiere kvantitative mål til resultatene produsert i respons av en mengde sangtekster, også omtalt som test-spørringer. Måten målet er komponert på er ved å sammenligne resultatene produsert av gjenfinningsmetodene opp mot menneskelig produserte resultater for de samme sangtekstene. Videre i seksjonen presenteres fremgangsmåten, resultatet og valg av gjenfinningsmetode.

5.1.1 **Fremgangsmåte**

På grunnlag av prosjektet sin størrelse har det ikke vært mulig å produsere menneskelige relevante resultater for alle 2, 745 unike sanger som inneholder sangtekst. Derfor er det blitt valgt en evalueringsmetode som ikke krever at alle relevante dokumenter er tilgjengelig. De to mest kjente målene å evaluere gjenfinningsmetoder på er precision og recall. Precision er et mål av evnen til en gjenfinningsmetode å returnere relevante dokumenter, mens recall er et mål av evnen dens til å returnere alle relevante dokumenter(Baeza-Yates & Ribeiro-Neto, 2011, s. 135).

I systemet brukes gjenfinningsmetodene til å presentere maksimum de fem mest relevante sangene for en gitt sang sin sangtekst. Dette er på grunn av at resultatene brukes i en force-directed graf, se seksjon 4.10.2, hvor det ikke er ønskelig å gjøre visualiseringen uoversiktlig ved å presentere flere

enn fem relaterte sangtekster. Det antas også at brukere ønsker at hvert resultat presentert vil være relevant (høy precision) istedenfor å presentere alle relevante resultater(høy recall) som oftest fører til lav precision. På bakgrunn av dette er precision vurdert høyere enn recall i systemet.

Målet som benyttes for å evaluere gjenfinningsmetodene er derfor average precision@5 for x antall sammenligninger/spørringer. Precision@5 er et mål for å beregne precision etter fem sangtekster er hentet fra resultatet til gjennfinningsmetoden. Målet reflekterer systemets ytelse som en bruker kan se (Baeza-Yates & Ribeiro-Neto, 2011, s. 140).

Etter Precision@5 er utregnet for alle spørringer, blir Average precision@5 regnet ut ved å summere Precision@5 resultatene og dele på x antall spørringer. Average precision@5 er valgt som metode for evaluering av gjenfinningsmetodene ettersom det reflekterer bruken av gjenfinningsmetodene i systemet. Resultatet av metoden gir derfor et reliabelt mål på om brukere av systemet vil bli presentert med relevante sanger for en gitt sang eller ikke.

For å bruke average precision@5 til å få et mål av hvordan de forskjellige gjenfinningsmetodene presterte ble det valgt ut 50 test-spørringer(sanger) som de ble testet på. Manning & Raghavan (2009, s. 140) presiserer at 50 test-spørringer er et representativt nummer for å få en god oversikt over hvordan gjenfinningsmetodene presterer. Test-spørringene brukt i evalueringen er 50 utvalgte sanger som har gode resultater for begge gjenfinningsmetodene, LSI og TF-IDF. Det vil si at en gitt test-spørring har resultater som har høy cosinuslikhet både for LSI og TF-IDF. Å velge 50 test-spørringer som oppfyller dette kriteriet gjør det mulig å teste gjenfinningsmetodene på en rettferdig måte, ettersom begge gjenfinningsmetoder har resultater som foreslår en viss likhet. Disse 50 sangene vil videre i seksjonen omtales som test-spørringene.

For hver av de 50 sangene i test-spørringene er det manuelt vurdert hvilke sanger returnert fra gjenfinningsmetodene som er relevante eller ikke. Denne vurderingen er gjort ved å lese og prøve å forstå alle sangtekstene som sammenlignes, for å så kunne avgjøre om de omhandler det samme emnet eller sender det samme budskapet. Et eksempel på dette er sangene ”Hero” av Mariah Carey og ”Beautiful” av Christina Aguilera som har forskjellige sangtekster, men begge sender det samme budskapet om at man skal elske seg selv uansett. Å tolke sangtekster åpner for menneskelige ”feil”, slik som å mistolke sangteksten sin betydning. Dette anses ikke å være et problem i evalueringen, ettersom sanger som regel er kryptiske og åpne for tolkning. For å prøve å sikre evalueringen mot forskjellige tolkninger av sangtekster er det derfor kun brukt en person til å tolke alle sangtekstene. Dette er gjort på grunnlag av antagelsen om at en person med sin bakgrunn og forståelse vil tolke sangtekster med lik betydning likt.

I tillegg til å evaluere TF-IDF er det valgt å evaluere fire versjoner av LSI med forskjellige k-

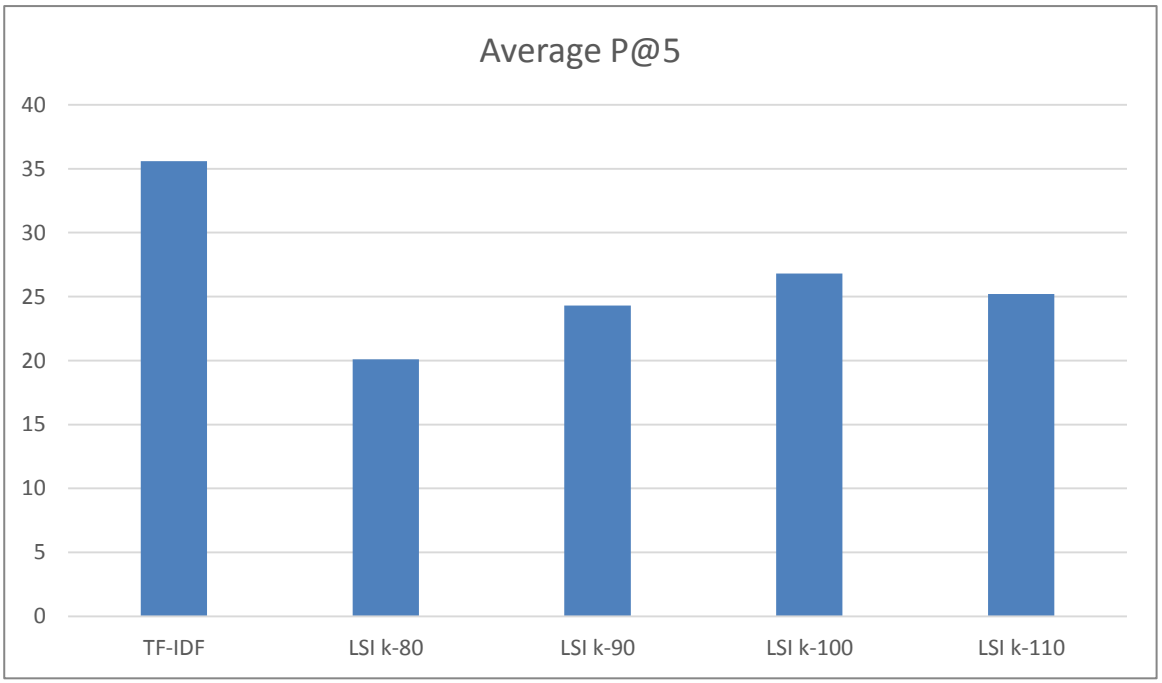
verdier. Dette er fordi verdien av k er avhengig av matrisen som skal reduseres og vil være viktig for at resultatene generert av LSI vil være relevante. Å velge antall dimensjoner, k, å redusere SVD-matrisene med er en utfordring, ettersom en reduksjon i k kan fjerne en stor del av støyen, men å ha for få dimensjoner kan gjøre at man mister viktig informasjon. I arbeidet gjort av Deerwester et al.(1990, s. 5) er det brukt 100 som en verdi for k, men Berry, Dumais, & O'Brien (1995, s. 20) har funnet ut at LSI ytelsen kan forbedres betraktelig etter en forskjell i k på 10 eller 20 dimensjoner og topper vanligvis mellom 70 og 100 dimensjoner, for å så begynne å avta langsomt. På grunnlag av disse resultatene ble LSI testet med en k-verdi på 80, 90, 100 og 110.

5.1.2 **Resultater**

	TF-IDF	LSI k-80	LSI k-90	LSI k-100	LSI k-110
Antall sanger testet	50	50	50	50	50
Antall unike ord	19, 175	19, 175	19, 175	19, 175	19, 175
Totalt sanger med sangtekst	2, 745	2, 745	2, 745	2, 745	2, 745
Average Precision@5	35,6	20,1	24,3	26,8	25,2

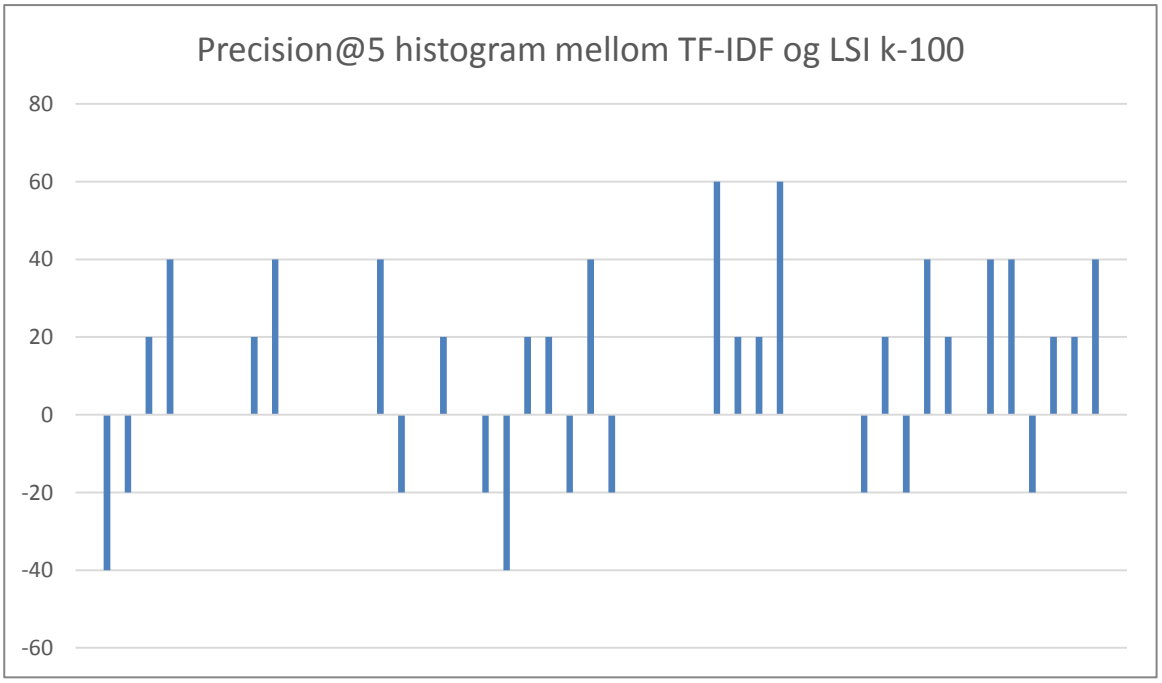
Tabell 5.1: Resultatene av evalueringen

I Tabell 5.1 presenteres resultatene av evalueringen. Evalueringen tilsier at Average Precision@5 for TF-IDF er bedre enn LSI k-100 med 8,8%. I Figur 5.1 er resultatene presentert i et stolpediagram for å få en visuell representasjon.



Figur 5.1: Average Precision@5 evaluering

For å se nøyere på resultatene mellom de to gjenfinningsmetodene som presterte best, TF-IDF og LSI k-100, er deres resultater over de 50 test-spørringene blitt sammenlignet ved å presentere de i et precision@5 histogram, se Figur 5.2.



Figur 5.2: Precision@5 histogram for de 50 test-spørringene mellom TF-IDF og LSI k-100

I Figur 5.2 er det presentert precision@5 av de 50 test-spørringene mellom TF-IDF og LSI k-100. Det er TF-IDF sett opp mot LSI k-100, derfor indikerer en positiv verdi i Figur 5.2 et bedre resultat for TF-IDF og en negativ verdi indikerer bedre resultat for LSI k-100. Figuren gir en rask oversikt over kvaliteten av resultatene til de to metodene og gjør det mulig å sammenligne visuelt. Man kan se at TF-IDF presterer bedre i 21 av test-spørringene, i 19 av test-spørringene presterer de likt, mens LSI k-100 kun presterer bedre i 10 av test-spørringene.

Resultatene var noe skuffende, ettersom det var forventet at LSI skulle prestere bedre enn TF-IDF. Denne antagelsen bygger på at LSI støtter sammenligning av dokumenter på basis av et konseptuelt tema eller betydningen av et dokument. LSI sammenligner derfor dokumenter ved den semantiske underliggende betydningen, noe som jeg antok ville vært interessant for sanger. Ettersom sanger inneholder flere måter å uttrykke et gitt konsept på(synonym) og ord ofte har flere betydninger(polysem).

LSI k-100 og TF-IDF resultatene er overlappende på flere av test-spørringene og det er som regel disse resultatene som er relevante for test-spørringen. Begge metoder viser å være bedre enn den andre i gitte situasjoner, men i flere resultater hvor de overlapper er det ofte at resultatene produsert av LSI k-100 ikke er relevante. Noen observasjoner hvor gjenfinningsmetodene returnerte et ikke-relevant resultat blir presentert videre.

LSI k-100 sine resultater for sangen “A Sky Full of Stars – Coldplay” med engelsk sangtekst resulterte i at en sang med norsk sangtekst, “Hodet over vannet - Prepple Houmb/Morten Abel”, var rangert øverst. Dette skjedde også motsatt vei, hvor fire av fem resultater for sangen “Vannski - Broiler” med norsk sangtekst var sanger med engelsk sangtekst. Som nevnt i seksjon 4.5 inneholder sangtekstene forskjellige språk, hovedsakelig engelsk, norsk og svensk. En mulig forklaring på de feil-produserte resultatene generert av LSI k-100 kan være på grunn av at LSI kun representerer hvert ord som ett punkt i det “semantiske rommet”(Deerwester et al., 1990, s. 21).
Det vil si at ord med flere helt forskjellige meninger, som “form“, blir representert som den gjennomsnittlige meningen av de forskjellige betydningene. Når datasettet i dette prosjektet inneholder ord som har forskjellige betydninger på forskjellige språk og muligens flere betydninger innenfor et språk vil den virkelige betydningen antageligvis være forskjellig fra den gjennomsnittlige betydningen. Dette generer resultater som reduserer kvaliteten av LSI og kan føre til at en leksikalsk tilnærming som TF-IDF vil prestere bedre.

Videre har resultatet for test-spørringen “Stay High - Tove Lo”, som omhandler det å være høy på

rusmidler, resultert i favør TF-IDF. LSI k-100 produserer flere irrelevante resultater som inneholder ordet "High" i meningen høy som i høyde eller høyere på "rangstigen", slik som i sangen "She's So High - Kurt Nilsen".
TF-IDF resultatene for lik test-spørring inneholder en relevant sangtekst av de fem returnerte. Den relevante sangteksten returnert er sangen "Because I Got High – AFROMAN", hvor ordet "High" blir brukt som høy på rusmidler og stemmer overens med betydningen i test-spørringen "Stay High - Tove Lo".
Som nevnt tidligere finner jeg disse resultatene forvirrende, ettersom jeg har antatt at akkurat slike test-spørringer ville produsere resultater i favør LSI. Det viser seg at resultatene strider imot LSI, ettersom ord som har flere betydninger(polysem) skal LSI være bedre til å ta hensyn til enn TF-IDF. Dette er kun et engangstilfelle og det konkluderes ikke med at LSI ikke fungerer slik det skal, men det er likevel en viktig observasjon.

5.1.3 **Valg av gjenfinningsmetode**

På bakgrunn av diskusjon og resultatene presentert ovenfor er det klart at TF-IDF er den dominerende gjenfinningsmetoden for sangtekster basert på systemet sitt datasett og bruksmåte. Grunnen til at akkurat disse gjenfinningsmetoden ble evaluert er på grunn av deres forskjellige tilnærminger. TF-IDF som har en leksikalsk tilnærming, mens LSI prøver å finne den "skjulte" semantikken og meningen bak teksten. Som nevnt tidligere ble det antatt at LSI ville prestere bedre enn TF-IDF i systemet, men dette var ikke tilfellet.

En utfordring for LSI som kan ha vært utslagsgivende for resultatet av evalueringen er problemet med å bestemme optimale antall dimensjoner som skal brukes for å redusere SVD, altså k-verdien. Færre dimensjoner tillater en bredere sammenligning av konseptene i sangtekstene, mens et høyere antall dimensjoner gir mer spesifikke sammenligninger av konsepter. På bakgrunn av ønsket om høy presisjon er det utprøvd relativt høye k-verdier i dette prosjektet. Det er mulig at verdien ikke er høy nok, slik at det generer en bredere sammenligning av konseptene i sangtekstene, noe som fører til høyere recall og lavere precision. Det er derimot vist til tidligere gode resultater med en k-verdi som er lik som i dette prosjektet. Deerwester et al. (1990) utførte LSI med gode resultater hvor k-verdien var 100 på CISI-settet som har 1460 dokumenter og på MED-settet som har 1033 dokumenter (Deerwester et al., 1990, s. 13, s. 16). Datasettet i denne evalueringen har 2, 745 dokumenter som inneholder sangtekster. På grunn av at systemet har flere dokumenter enn i

Deerwester et al. (1990) sine eksperimenter skulle man antatt at k-verdien burde ha blitt økt enda mer, men som illustrert i Figur 5.1, avtar precision når k-verdien økes fra 100 og til 110.

Som nevnt i resultatene er det kun LSI som har foreslått at en engelsk sangtekst er lik en norsk sangtekst og motsatt. En alternativ løsning som kunne resultert i at LSI ville prestert bedre, hadde vært å analysere hvilket språk de forskjellige sangtekstene inneholdt, slik at sangtekstene kunne blitt kategorisert ut ifra språket de tilhører. Dette ville fjernet muligheten for at en sang med engelsk sangtekst ville vært lik en sang med norsk sangtekst ved bruk av LSI.
Uansett, på bakgrunn av antagelsen om at flere språk gir LSI problemer, så ville det vært interessant å prøve gjenfinningsmetodene på et "nøytralt" datasett som ikke inneholder flere forskjellige språk, for å se hvordan de ville prestert.

Som resultatene tilsier er det klart at TF-IDF presterte bedre enn LSI i dette systemet og er derfor den gjenfinningsmetoden som brukes i systemet. Det faktum at dens resultater var 8,8% mer presise støtter avgjørelsen.

5.2 Usability-testing

Denne seksjonen tar for seg usability-testingen av web-applikasjonen. Usability er oftest definert som brukervennligheten og aksepten av et system for en spesiell klasse av brukere som utfører bestemte arbeidsoppgaver i et bestemt miljø. Brukervennligheten påvirker brukerne sin tilfredshet og deres ytelse, mens aksept påvirker om produktet vil bli brukt (Bevan, 1995, s. 2).

Presentasjonen av usability-testingen er basert på formatet foreslått av Dumas & Redish (1999), slik at andre enkelt kan avgjøre validiteten og reliabiliteten av resultatene presentert. Formatet presentert av Dumas & Redish (1999, s. 349) er basert på en rapport om usability-testing. Derfor er formatet som benyttes i denne avhandlingen tilpasset, ved å fjerne sammendrag og vedlegg.

Det tilpassede formatet inneholder følgende seksjoner:

- Mål
- Metoder
- Prosedyre
- Resultater

- Forklaring av funn og anbefalinger

5.2.1 **Mål**

Usability-testen hadde en rekke mål.

Det første målet var et generelt mål om å finne ut om web-applikasjonen er brukervennlig.

Det andre målet var å forstå hvordan brukerne brukte applikasjonen når de fikk frihet til å eksperimentere fritt. Observasjoner fra denne typen "sandkassemodus" hjalp til å trekke visse konklusjoner om hvor attraktivt web-applikasjonen var og bruken dens.

Det tredje målet var å vurdere om brukerne ville forstå de data og visualiseringer som blir presentert til dem. Kanskje visualiseringene ville være for komplekse og ikke formidle informasjon til brukeren. Dette målet var også spesielt rettet mot de akustiste egenskapene til sangene og om brukere forstår betydningen av de, for eksempel dansbarhet og skala. For å oppnå dette, var det viktig å lage scenarioer hvor visualiseringer og akustiske egenskaper ble presentert.

Det fjerde målet var å finne ut hvorvidt brukere klarte å navigere seg frem til ønsket informasjon. Dette ville kreve at brukerne utnyttet det de har lært fra bruken av web-applikasjonen til å navigere seg frem.

5.2.2 **Metoder**

Metodene brukt i usability-testingen ble valgt slik at det ville være mulig å besvare målene.

For å få en generell oversikt over web-applikasjonen sin brukervennlighet ble det gjennomført en System Usability Scale(SUS)-spørreundersøkelse. SUS er en enkel, tipunkts-spørreundersøkelse som gir en global oversikt over subjektive vurderinger av usability. Brukeren skal rangere 10 påstander om systemet på en Likert skala, fra helt uenig til helt enig (Brooke, 1996, s. 191).SUS skal brukes rett etter at deltakeren har hatt anledning til å bruke systemet, før debriefing eller diskusjonen finner sted. Dette er for at brukeren skal fylle ut sin umiddelbare reaksjon til hver påstand, istedenfor å tenke på påstander over lang tid. Alle påstandene skal utfylles og hvis en deltaker føler at de ikke kan svare på en bestemt påstand, skal de markere midtpunktet av skalaen.

Målet om hvordan web-applikasjonen brukes ble funnet ut ved å observere testdeltakere når de fikk

friheten til å eksperimentere fritt, i en såkalt "sandkassemodus". Mens testleder observerte deltakere bruke web-applikasjonen ble de bedt om å “tenke høyt”. Denne metoden kalles “Thinking Aloud” og gjør det mulig å forstå hvordan deltakere ser på web-applikasjonen, noe som gjør det lett å identifisere deltakernes store misforståelser (Nielsen, 1993, s. 195). Ifølge Nielsen (1993, s. 195) kan metoden være den mest verdifulle usability engineering metoden, ettersom den genererer store mengde kvalitative data ut fra et relativt lite antall brukere.

For å finne ut om om brukere ville forstå de presenterte data og om brukere klarte å navigere seg frem til ønsket informasjon ble testdeltakerne gitt scenarier, som beskrev konteksten bak hvorfor en bestemt bruker besøker web-applikasjonen og hva han ønsker å oppnå.

Kvantitative data ble innsamlet ved SUS-spørreundersøkelsen og deltakers ferdigstillelse(suksess eller fiasko) av scenarier.

Kvalitative data ble innsamlet gjennom Thinking Aloud metoden når deltaker testet web-applikasjonen.

Testdeltakere

Det var totalt åtte testdeltakere som gjennomførte usability-testingen, på onsdag 22. April. Syv av testdeltakerne var menn og en var kvinne. Deltakerne er i en aldersgruppe fra 20-30 år og har alle tidligere erfaring med bruk av datamaskin. Ingen av brukerne hadde derimot noen spesiell form for musikalsk bakgrunn. Dette var viktig for å kunne kartlegge om brukerne ville forstå hva som blir presentert til dem. Musikkeksperter vil antageligvis lettere forstå de akustiske egenskapene til en sang og det var derfor ikke ønskelig at de var testdeltakere.

Antall deltakere ble valgt på grunnlag av usability-testen sin reliabilitet. Nielsen(1994, s. 385) konkluderer med at en Usability-test for åtte personer med Thinking Aloud metoden vil finne 90-95% av alle problemene i et system.Videre presenterer Tullis & Stetson (2004) i sin forskning at bruk av SUS gjør det mulig å få et mål på den oppfattede usabilityen av et system med åtte testdeltakere og være ganske sikker at man har en god vurdering av hvordan folk ser systemet(75%) (Tullis & Stetson, 2004, s. 37). Antall testdeltakere er derfor bestemt for at resultatene skal være reliable, men også av hensyn til prosjektets tidsaspekt. For å kunne oppnå en 100% sikkerhet i resultatene, ville det vært nødvendig å teste på mange flere brukere, noe som ville tatt lang tid.

Miljø

Testmiljøet ble satt opp i et seminarrom på UIB, onsdag 22. April. Seminarrommet var stille og

isolert mot støy. Datamaskinen brukeren skulle utføre oppgaver på var en bærbar datamaskin med 1366x768 skjermoppløsning. Oppløsningen ble bestemt, ettersom dette er den mest brukte på Internett, ifølge statistikk presentert i Januar, 2015 av w3schools.com[19]. Datamaskinen ble plassert på et skrivebord slik at nødvendige instrukser ville være lett synlig når det ble plassert foran brukeren. Nettleseren web-applikasjonen ble presentert i var Google Chrome, versjon 42.0.2311.135. Det eneste kravet for valg av nettleser var at den skulle være moderne, altså ikke eldre enn Internet Explorer 10. Dette er på grunn av at enkelte front-end rammeverk benyttet i web-applikasjonen ikke støtter eldre nettlesere. Ved å benytte kun en nettleser var det også enkelt å lokalisere om en potensiell feil lå i web-applikasjonen eller i selve nettleseren.

5.2.3 **Prosedyre**

1. Testleder ønsker velkommen til testdeltaker og forklarer prosedyren, spør deltakeren om å signere et samtykkeskjema som inneholder informasjon om at hans handlinger vil bli dokumentert og brukt i prosjektet, samt at de vil forbli anonyme og at testen kan avbrytes hvis han ønsker, se Appendiks G Samtykkeskjema..
2. Testleder forklarer sesjonen, verktøy som skal brukes(bærbar datamaskin, notisblokk), metoder(SUS og Thinking Aloud) og spør om deltakeren har flere spørsmål. Testleder vil så starte testingen.
3. Deltaker blir bedt om å eksperimentere fritt i web-applikasjonen, mens han sier hva han tenker. Testleder observer deltakeren nøye i starten og over tid. De første minuttene var veldig viktige å observere nøye, ettersom det ga viktig informasjon om hvor enkelt web-applikasjonen var å lære seg å bruke.
4. Deltaker blir presentert med scenarioene, se Appendiks H Scenarier. Deltaker vil lese spørsmålet høyt før han starter å løse scenarioet, mens han sier hva han tenker. Hvis brukeren har problemer med å utføre scenarioet, skal det gis litt hjelp først, og deretter ytterligere hjelp om det er nødvendig. Tanken er at scenarioene skal være enkle å fullføre gjennom utformingen av selve grensesnittet til web-applikasjonen. Hvis deltakeren feiler scenarioet, så gis han en ny mulighet. Dette er for å samle inn mer data, for å kunne forstå hvorfor deltaker feiler.
5. Testleder deler så ut System Usability Scale-spørreundersøkelsen, ber deltaker om å utfylle

[19] http://www.w3schools.com/browsers/browsers_display.asp

sin umiddelbare reaksjon til hver påstand, istedenfor å tenke på påstander over lang tid. Hvis deltaker føler at han ikke kan svare på en bestemt påstand, skal de markere midtpunktet av skalaen.

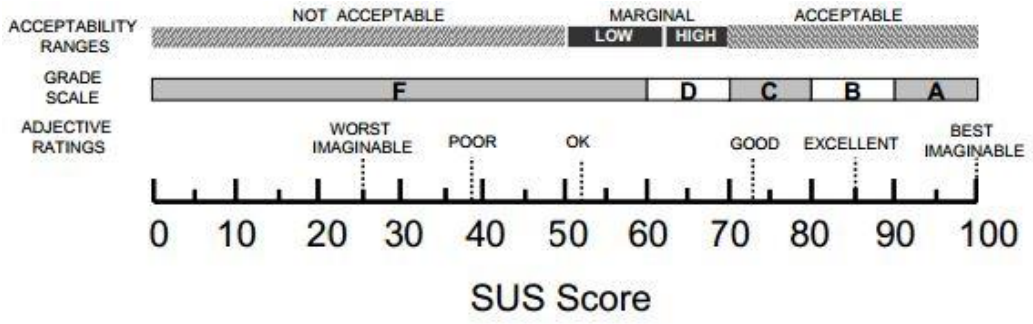
6. Ved endt test er det åpent for subjektive spørsmål fra begge parter og diskusjon, før testleder takker deltakeren og eskorterer han ut fra testmiljøet.

Analyse

For å finne prosent av deltakere som fullførte scenarioene og prosent av deltakere som ikke fullførte scenarioene er de kvalitative data innsamlet ved observasjon blandet med de kvantitative data for deltaker sin ferdigstillelse(suksess eller fiasko) av scenarier. Disse dataene er så blitt brukt til å gjennomføre en analyse av kilden til feilen (Source of Error Analysis). Rubin (1994, s. 276) omtaler en slik analyse som det ultimate detektiv arbeid, hvor det er meningen å finne system-relaterte grunner til testdeltakere sine feil eller dårlige prestasjoner. Gjennom denne analysen er de kvalitative data innsamlet om deltakeres handlinger nøye analysert, for å kunne påpeke feilkilden i web-applikasjonen og hvilket komponent som er "skyldig".

SUS-spørreundersøkelsen ga et enkelt tall som representerer et sammensatt mål av den samlede brukbarheten til systemet. For å beregne SUS poengsummen, ble verdien fra hvert element i spørreundersøkelsen summert. Hvert element sin verdi vil variere fra 0 til 4. For elementer 1,3,5,7 og 9 vil verdien være skalaen sin posisjon minus 1. For elementer 2,4,6,8 og 10, er verdien 5 minus skalaen sin posisjon. Tilslutt ble summen av verdiene multiplisert med 2,5 for å få den samlede verdien av SU(System Usability)(Brooke, 1996, s. 5).

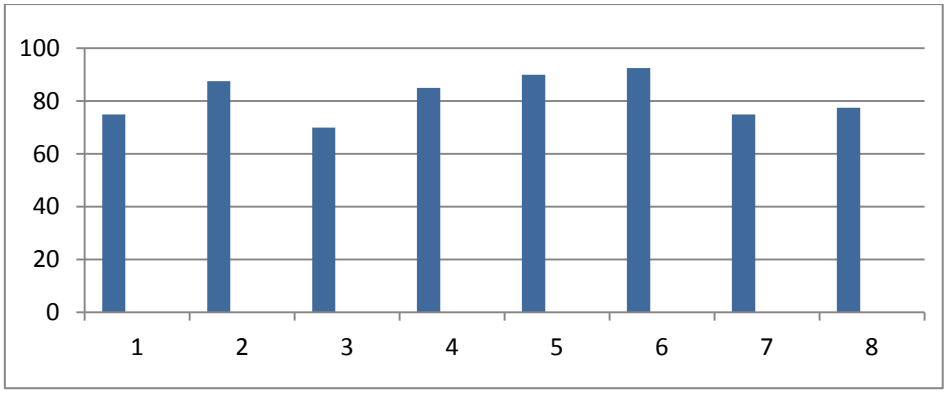
SUS poengsummen har en verdi fra 0 til 100 og det har i lengre tid vært en usikkerhet rundt hva som er en god verdi. I en forskningsrapport av Sauro(2011a) har det blitt regnet ut SUS poengsum for 500 undersøkelser hvor gjennomsnittet var 68 (Sauro, 2011a, s. 36). Bangor, Kortum, & Miller (2009) regnet ut gjennomsnittet for 3, 500 undersøkelser og fikk et resultat på 70 (Bangor et al., 2009, s. 117). På bakgrunn av disse resultatene presenterer Bangor et al. (2009) en mulig måte å tolke en SUS poengsum på, se Figur 5.3.



Figur 5.3: Gradering av SUS score, fra Bangor et al. (2009, s. 121)

5.2.4 **Resultater**

SUS(System Usability Scale)

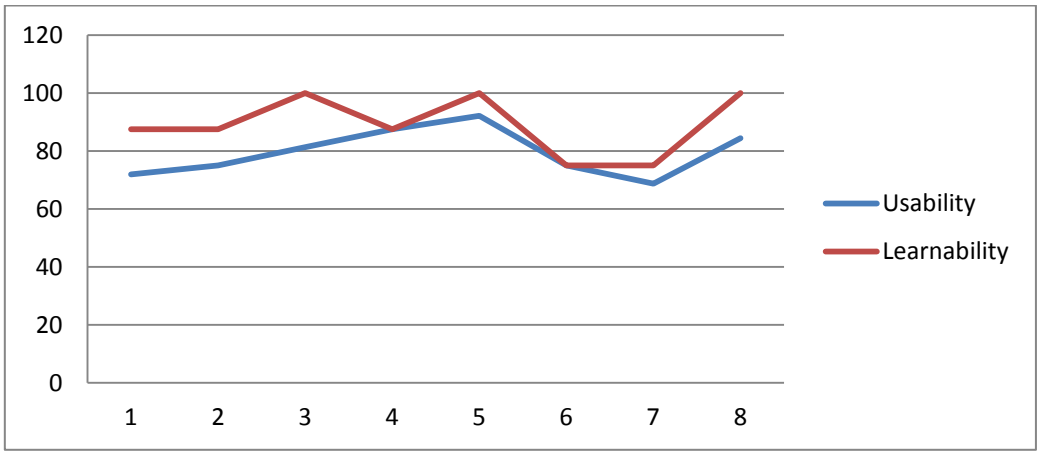


Figur 5.4: Resultater av SUS-undersøkelsen

Figur 5.4 viser de forskjellige SUS poengsummene for de 8 SUS-spørreundersøkelsene som ble utført. Den gjennomsnittlige SUS poengsummen var 81,65. Denne poengsummen rangeres som akseptabel og god, ifølge Bangor et al. (2009) og Figur 5.3.

For å trekke ut ytterligere informasjon fra den samlede SUS poengsummen er det ifølge Lewis & Sauro (2009) sin forskning, mulig å dekomponere samlet SUS poengsum inn i to komponenter: usability(brukervennlighet) og learnability(lærbarhet). Dekomponeringen baserer seg på at spørsmål 4 og 10 i SUS-spørreundersøkelsen omhandler learnability og de resterende 8 spørsmålene omhandler usability. Denne dekomponeringen gjør det mulig å se på de to komponentene hver for

seg. Resultatet av dekomponeringen vises i Figur 5.5.

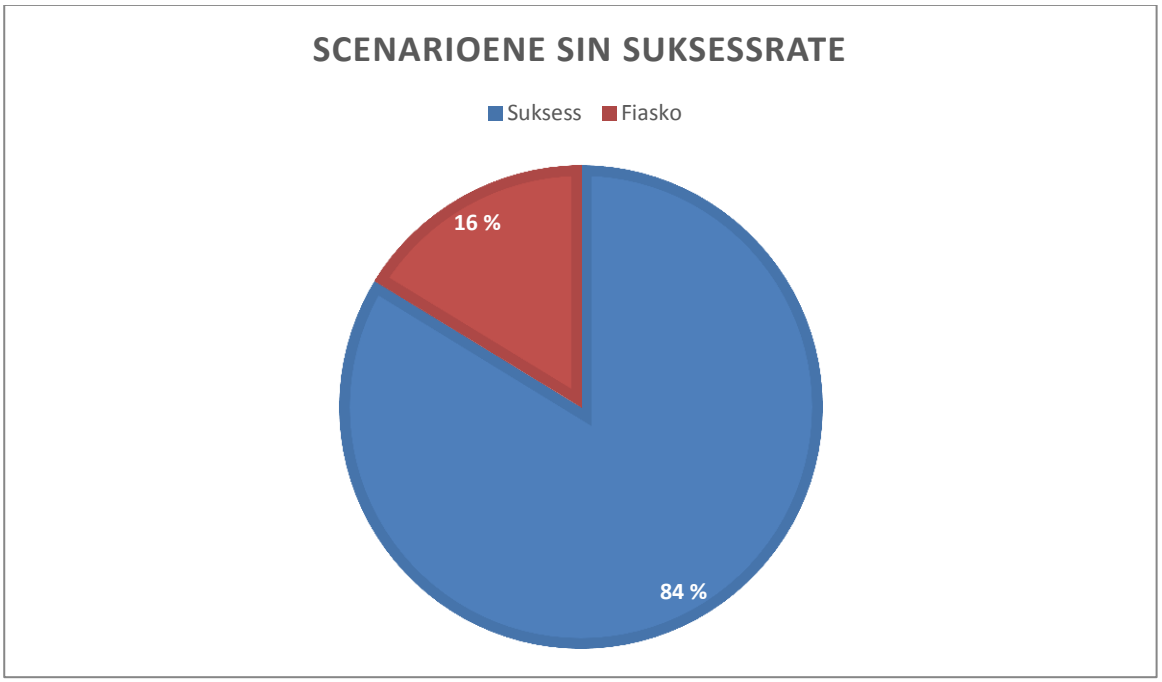


Figur 5.5: Dekomponering av SUS poengsum

I Figur 5.5 kan man se en moderat korrelasjon mellom usability og learnability. Når verdien av usability synker, har også verdien av learnability en tendens til å avta. Tendensen som vises er at verdien av learnability er for det meste større enn verdien av usability, men i varierende grad. Dette stemmer overens med Lewis & Sauro (2009, s. 8) sine resultater og gjør det mulig å få et anslag på oppfattet learnability sammen med et bedre estimat av den oppfattetede usabilityen. Den gjennomsnittlige verdien for learnability er 89 og den gjennomsnittlige usabilityen er 79.5.

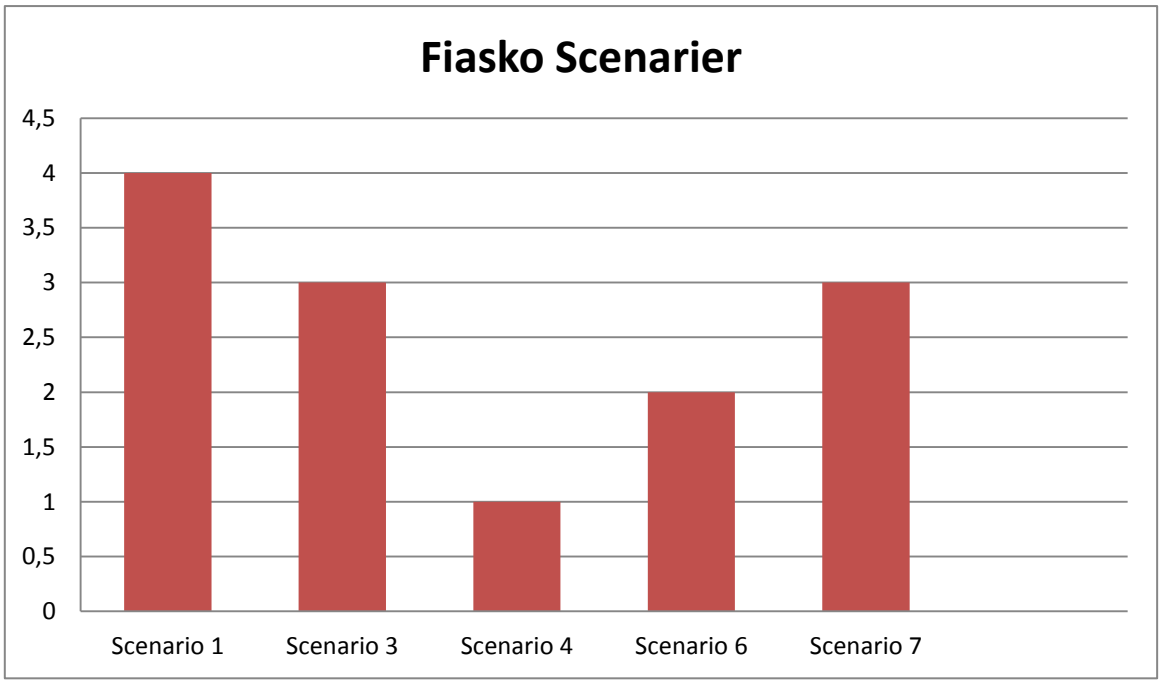
Scenarier

Av de 80 scenariene gjennomført endte 67(84%) i suksess og 13(16%) i fiasko.



Figur 5.6: Scenarioene sin suksessrate

Figur 5.6 viser en representasjon av antall scenarier som endte i suksess eller fiasko. Videre er det interessant å analysere de scenarioene som feilet, for å kunne gjennomføre en kilde av feil analyse(Source of Error Analysis) for å finne ut hva som førte til at testdeltakerne feilet. Fordelingen av de 13 scenarioene som endte i fiasko er representert i Figur 5.7



Figur 5.7: Fordeling av scenarier som endte i fiasko

Fordelingen av de 13 scenarioene som endte i fiasko(16%) er ikke veldig spredt. Det er hovedsakelig de samme scenarioene som feilet flere ganger, med unntak av scenario 4, som kun feilet 1 gang (8%). Scenario 1 endte i fiasko 4 ganger og står for 30% av de ikke fullførte scenariene. Scenario 3 og 7 endte begge i fiasko 3 ganger(23%), mens scenario 6 kun endte i fiasko 2 ganger(15%)

Source of error analyse

Scenario	Kilde til feil:
Scenario 1: Du jobber som musikkjournalist og er interessert i utviklingen av lydstyrke i norsk popmusikk.	• Tre testdeltakere lette etter en link til sammendrag eller år i menyen. De skulle ha trykket på liste-linken, men brukere fant ikke dette intuitivt. • En testdeltaker fant ikke dropdown-listen slik at han kunne sortere linjegrafen etter lydstyrke.
Scenario 3: Du er uenig i en venn sin påståelse om	• Tre testdeltakere søkte opp artisten Madonna og fant data om antall ganger artisten var listet, sett opp mot gjennomsnittet,

at den mest listede pop-artisten i Norge de siste 50 årene er Madonna og ønsker å finne svar på denne påståelsen.	men fant ikke ut om hun var den artisten som var listet flest ganger. Testdeltakeren skulle ha besøkt artister-siden og sortert etter antall ganger listet.
Scenario 4: Du er en energisk person som liker nyere sanger(etter år 2000) med høyt tempo og masse energi. Du ønsker å oppdage flere slike sanger.	• En testdeltaker forsto ikke den grafiske framstillingen av resultatet i et spredningsplott. Skjønte ikke at et objekt med høy x-verdi og y-verdi ville være lokalisert i top høyre hjørnet i spredningsplottet. Det kan diskuteres om feilen ligger hos testdeltaker, på grunn av manglende kunnskap om spredningsplott/grafer eller om et spredningsplott er dårlig til å presentere disse data.
Scenario 6: Du har hørt på favorittartisten din, Jim Reeves, i mange år, men det er få personer du møter som liker musikken hans. Du undrer deg selv om artisten Jim Reeves er listet mange ganger i Norsk musikk sin historie.	• To testdeltakere besøkte artisten sin side for å lete etter informasjon om antall uker artisten hadde vært på listen, men fant ikke svaret blant de mange data presentert på siden. Dataen de var på jakt etter er presentert under "Lyd-kategorien". Deltakerne fant det ikke intuitivt å skulle se under "Lyd"-kategorien for artisten, for å finne ut antall ganger artisten har vært listet på VG-Lista.
Scenario 7: Du er usikker på når Michael Jackson sin sang Thriller ble utgitt og ønsker å finne ut av dette.	• To testdeltakere brukte søkemotoren på siden for å søke etter sangen. Deltakere forventet at "søke-knappen" skulle føre de til en ny side med søkeresultat, noe som ikke er tilfelle, ettersom de skulle ha valgt resultat fra typeaheaden når de skrev inn søkeord. Dette førte til at Scenarioet stoppet ved søk. • En testdeltaker som klarte å navigere seg inn på sangen sin side hadde problemer med å finne svaret, på grunn av de store mengdene av data presentert til dem.

5.2.5 **Forklaring av funn og anbefalinger**

Denne seksjonen vil gjennomgå funnene fra brukertestingen og se de opp mot målene av testen:

- Er web-applikasjonen brukervennlig?
- Hvordan brukes web-applikasjonen?
- Forstår brukere de data og visualiseringer som blir presentert?
- Klarer brukere å navigere seg frem til ønsket informasjon?

Videre vil det presenteres anbefalte endringer i web-applikasjonen, for å forbedre brukervennligheten.

Det første målet, å finne ut om web-applikasjonen var brukervennlig, er blitt undersøkt gjennom analyse av både kvalitative og kvantitative data innsamlet fra brukertesten. De kvalitative data innsamlet ved å observere testdeltakere bruke web-applikasjonen ga innsikt i applikasjonen sin brukervennlighet. Under observasjonen viste testdeltakere aldri tegn til frustrasjon, selv om de til tider ikke fullførte use-casene. Ved anledninger var deltakere usikre på hvordan de skulle navigere seg frem og finne svar på use-casene. Dette er noe som er nødt til å bli tatt tak i for å øke brukervennligheten. Generelt sett viste observasjoner av deltakere at web-applikasjonen har en god brukervennlighet. For å få et bedre svar på om observasjonene er rett, gjennomførte testdeltakerne SUS-spørreundersøkelsen, hvor de selv vurderte web-applikasjonen sin brukervennlighet. Resultatet av SUS-spørreundersøkelsen produserte en SUS-poengsum på 81, 65. Poengsummen rangeres som akseptabel og god, ifølge Figur 5.3, og styrker observasjonen om at web-applikasjonen har en god brukervennlighet.

Det andre målet, hvordan web-applikasjonen brukes, ble undersøkt ved å observere testdeltakere sine handlinger når de fikk frihet til å eksperimentere fritt i web-applikasjonen. Seks av åtte deltakere søkte umiddelbart opp en av sine favorittartister eller favorittsanger for å utforske. De resterende to deltakerne besøkte “Data beskrivelse”-siden, før de klikket seg videre i web-applikasjonen. Under observasjonen trengte ingen av deltakerne noen form for hjelp til å komme i gang, noe som er veldig positivt og viser at deltakerne var interessert i å utforske hva web-applikasjonen har å tilby. Dette skyldes nok at applikasjonen omhandler musikk, og at de fleste personer i dagens samfunn har et forhold til musikk.

I løpet av sesjonen utprøvde deltakere de forskjellige funksjonene i web-applikasjonen og det så ut til at de fant visualiseringen på “oppdag”-siden mest interessant. Flere av deltakerne brukte flere minutter på å oppdage forskjellige objekter ved å endre filtre som akustiske egenskaper og år. Under

bruken av "oppdag"-siden ble det observert flere positive reaksjoner når web-applikasjonen sin presentasjon av et objekt var likt med brukeren sin oppfatning. Slik som at artistene Scooter og E-type hadde høy energi.

På grunnlag av observasjon av testdeltakere sine handlinger kan det sies at web-applikasjonen ble brukt til å oppdage og utforske musikk, og at testdeltakere fant det interessant.

Det tredje målet var å vurdere om brukerne vil forstå de data og visualiseringer som blir presentert til dem. Gjennom observasjon av testdeltakere sine handlinger og reaksjoner ved bruk av web-applikasjonen har det vist seg at ikke alle brukere forstår all data de blir presentert. En deltaker forstod ikke spredningsplottet på "Oppdag"-siden, som blir brukt til å visualisere forskjeller mellom objekter. Dette betyr ikke nødvendigvis at spredningsplottet er en dårlig visualisering for slike data, ettersom man skulle anta at brukere vet hvordan grafer fungerer. Å vite forskjellen på x og y-aksen og hvor høye og lave verdier vil plasseres på disse aksene. Uansett så er det nevneverdig, ettersom web-applikasjonen har som mål å være brukervennlig for alle brukere. Resterende testdeltakere forstod visualiseringene og samhandlet med de, ved å endre akustiske egenskaper og filtrere etter år.

Når deltakere ble presentert med flere data og visualiseringer viste det seg at de ikke alltid fant frem til sitt informasjonsbehov. Tre av deltakerne fullførte ikke use-casene på grunn av dette. En mulig forklaring på problemet er at brukere blir presentert med for mye data som ikke er kategorisert eller strukturert på en måte de finner logisk, også kalt "information overload". Observasjonene resulterte i at de fleste brukere forstod de data og visualiseringer som ble presentert, men at det er noen problemområder som kan forbedres.

Det fjerde og siste målet var å finne ut hvorvidt brukere kan navigere seg frem til ønsket informasjon. Dette omfatter bruken av navigasjonen og søkemotoren. Fire av testdeltakerne uttrykte usikkerhet ved bruk av søkemotoren. De trykket på noe som kan ligne en "søkeknapp" og forventet å bli omdirigert til en side med resultater for søket. "Søkeknappen" var kun ment som en indikasjon på at feltet er et søkefelt, og har ingen funksjon ved klikk. Deltakerne skulle ha trykket på et element i typeahead-listen etter de hadde begynt å skrive, for å så bli omdirigert til siden for objektet.

Ved videre observasjon kommenterte fire brukere at det ikke var særlig intuitivt å trykke på "liste"-elementet når de letet etter informasjon for norsk popmusikk sin historie. Navigasjon/Navigering mellom de andre elementene i navigasjonen fungerte bra og det så ut til at deltakere fant elementene beskrivende for sitt informasjonsbehov.

For å oppsummere mål fire, så klarte brukere i stor grad å bruke navigasjonen til å navigere seg frem til ønsket informasjon, men fant det ikke logisk for sitt informasjonsbehov å trykke på "liste"-

elementet. Søkemotoren utløste frustrasjon og usikkerhet hos brukeren, og fungerte ikke slik som tiltenkt.

På bakgrunn av observasjoner gjort av testdeltakere sine feil og source of error analysen er det laget en liste med anbefalte endringer i web-applikasjonen, som kan forbedre brukervennligheten.

Anbefalinger i prioritert rekkefølge	
1.	Anbefaling: Endre navn fra "Liste" og til "År" i menyen
	Forklaring: Tre testdeltakere fant ikke utviklingen av lydstyrke i norsk popmusikk, på grunn av lite beskrivende navngiving i menyen. Fire testdeltakere kommenterte at det ikke var særlig intuitivt å trykke på liste når de lette etter informasjon for norsk popmusikk sin historie.
2.	Anbefaling: Implementere en rangering for hvor mange ganger artisten er listet.
	Forklaring: Tre testdeltakere fant ikke informasjon om hvor mange ganger en artist er listet i forhold til andre.
3.	Anbefaling: Gi søkeknappen en funksjon.
	Forklaring: To testdeltakere klarte ikke å bruke søkemotoren, ettersom de ikke trykket på et element i typeaheaden, men heller trykket på søkeknappen(som egentlig er en illustrasjon for å vise at brukere kan søke i feltet). Fire av testdeltakerne som klarte å bruke søkemotoren trykket først på "søke-knappen", før de skjønte at de måtte trykke på elementet i typeahead-listen.
4.	Anbefaling: Samle "generell informasjon" i en egen kategori
	Forklaring: En testdeltaker fant ikke informasjon om når en sang først ble listet. To testdeltakere brukte lang tid på å finne informasjonen, ettersom de mente det var "gjemt" i en visualiseringen. To testdeltakere fant ikke antall uker artisten hadde vært listen, på grunn av at informasjonen fantes i "Lyd-kategorien".

6 Konklusjon og fremtidig arbeid

Før arbeidet med avhandlingen startet fant jeg det interessant å kunne utforske og få en forståelse av komplekse musikkolleksjoner. Spesielt i denne tidsepoken hvor utviklingen av Internett, bærbare enheter, multimedia komprimering og lagringsteknologi har i stor grad økt kompleksiteten til både private og kommersielle musikksamlinger. For å ta utgangspunkt i en kompleks musikkolleksjon ble norsk popmusikk sin musikkolleksjon valgt. Dette var ikke kun på grunn av at jeg ønsket å utforske den selv, men også for at andre skal kunne ta nytte av det. Jeg tror at det å kunne presentere mønstre og relasjoner i norsk popmusikk sin historie kan være av interesse for musikkinteresserte personer, historikere, journalister, produsenter, låtskrivere, med mer.

På bakgrunn av dette ble et forskningsspørsmål formulert:

"Hvordan støtte gjenfinning og visualisering av relasjoner i norsk popmusikk?"

For å besvare forskningsspørsmålet har jeg utviklet og testet et system med fokus på datavisualisering og informasjonsgjenfinning.

I avhandlingen ble det implementert en søkerobot for å innhente nødvendige data om norsk popmusikk fra 1960-2014. Søkeroboten webskrapet først VG-Lista Topp 20 sine nettsider for data om lister og sanger, før den samlet inn komplementære data som sangtekst og akustiske egenskaper fra de eksterne APIene, Echonest og Musixmatch.

For å oppdage relasjoner mellom sangtekster ble det brukt to forskjellige gjenfinningsmetoder, termfrekvens-invers dokumentfrekvens(TF - IDF) og latent semantic indexing(LSI). TF-IDF, med sin leksikalske tilnærming som sammenligner dokumenter på ord og antar at de betyr det samme i de forskjellige dokumentene, og LSI med sin semantiske tilnærming, som muliggjør sammenligning av dokumenter på bakgrunn av et konseptuelt tema eller betydningen av et dokument.

Begge gjenfinningsmetodene ble implementert i systemet før de ble evaluert for å finne ut hvilken av de to som ga mest presise resultater for de innsamlede dataene.

En egendesignet algoritme ble implementert i systemet for å oppdage relasjoner mellom sanger basert på akustiske egenskaper.

En web-applikasjon ble laget for å presentere mønstre og relasjoner i norsk popmusikk. Data ble presentert i visuelle komponenter slik at brukere fikk en god innsikt i dataene og muligheten til å trekke konklusjoner, ved å direkte samhandle med data.

Web-applikasjonen ble så evaluert gjennom en usability-test ved å teste applikasjonen på brukere.

6.1 Oppsummering av funn

Jeg vil nå oppsummere resultatene av evaluering av gjenfinningsmetodene og usability-testingen av web-applikasjonen gjort i kapittel 5.

Evalueringen av gjenfinningsmetodene resulterte i at TF-IDF presterte bedre enn LSI for sammenligning av sangtekster i systemet. Det er viktig å presisere at resultatet av evalueringen kun sier noe om hvordan gjenfinningsmetodene presterte i systemet og ikke på et overordnet nivå. Forskjellige grunner til hvorfor LSI presterte dårligere enn TF-IDF er blitt diskutert. En grunn kan være at systemet inneholder sangtekster fra forskjellige språk. Dette er et problem for LSI, ettersom ord med flere betydninger blir representert som den gjennomsnittlige betydningen av de forskjellige betydningene. Så når systemet inneholder ord som har forskjellige betydninger på forskjellige språk og muligens flere betydninger innenfor et språk vil den virkelige betydningen som oftest være forskjellig fra den gjennomsnittlige betydningen.
Dette har ført til at en av de store fordelene med LSI, altså støtte for synonymer og polysemer, har heller blitt en ulempe. Det er også diskutert muligheten for at antall dimensjoner LSI reduserer SVD-matrisen med er for lav til å produsere presise sammenligninger. Dette er sett opp mot tidligere forskning av Deerwester et al. (1990), men valget av en optimal *k*-verdi er i bunn og grunn unik for hvert system.

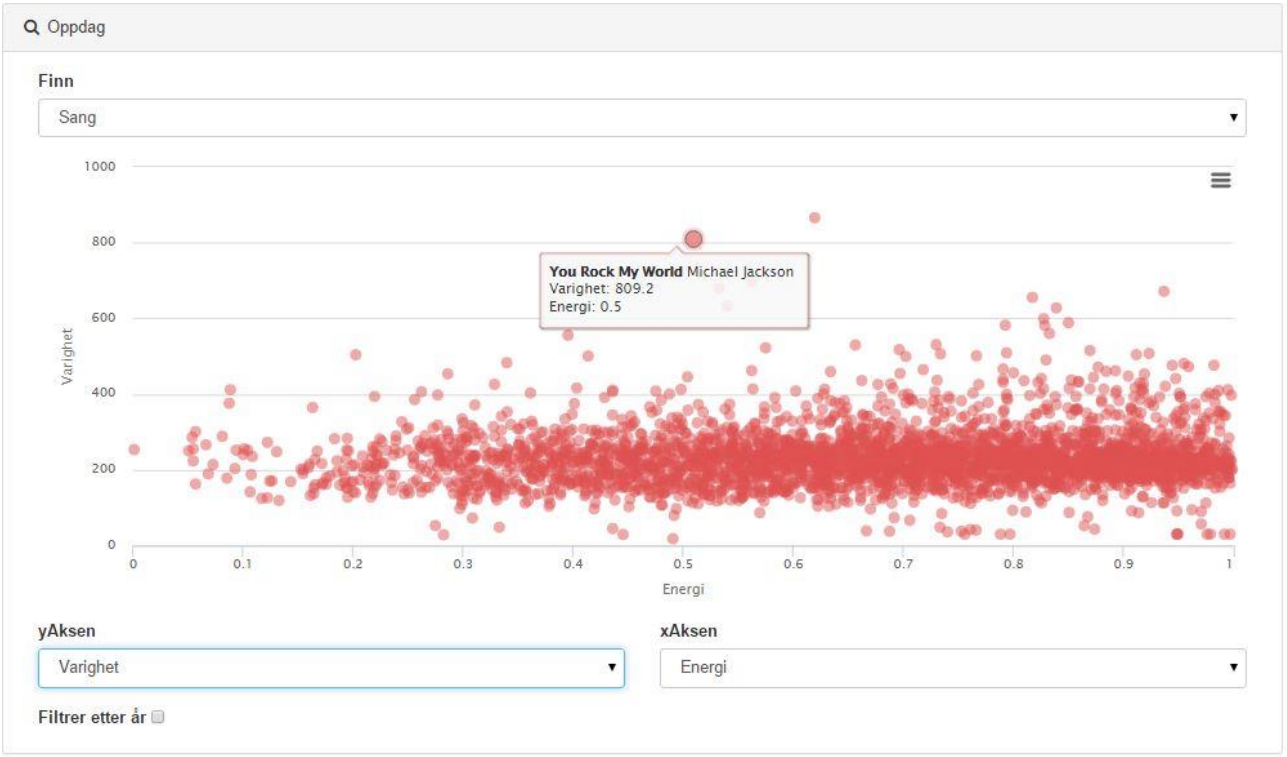
Usability-testingen ble utført på åtte testdeltakere med mål om å finne ut om web-applikasjonen var brukervennlig, å forstå hvordan brukerne bruker applikasjonen, å vurdere om brukerne vil forstå de data og visualiseringer som blir presentert til dem og å finne ut hvorvidt brukere kan navigere seg frem til ønsket informasjon. Brukertesten genererte kvalitative og kvantitative data som i stor grad støttet at brukervennligheten til web-applikasjonen var god. Den kvantitative SUS-spørreundersøkelsen resulterte i en gjennomsnittlig SUS-poengsum på 81,65. SUS-poengsummen rangerte brukervennligheten til web-applikasjonen som akseptabel og god, ifølge Bangor et al.(2009) og Sauro (2011) sine resultater. Videre ble det utført en "source of error"-analyse hvor de use-casene som feilet flere ganger ble nøye analysert for å kunne finne ut hvilke deler av web-applikasjonen som feilet. Analysen av de kvalitative data innsamlet fra brukere sine handlinger og reaksjoner, samt "source of error"-analysen resulterte i en liste med anbefalinger over hvilke komponenter i web-applikasjonen som bør endres/forbedres.

6.2 Refleksjon

Utviklingen av systemet, resultatene det presenterer og evalueringene er utført for å besvare problemstillingen, hvordan støtte gjenfinning og visualisering av relasjoner i norsk popmusikk. For å sikre at metodene har produsert pålitelige resultater som kan bli reprodusert er de blitt nøye evaluert og testet. I denne seksjonen vil det reflekteres over deler av avhandlingen som kan svekke systemet. Dette er mest relatert til innsamling av data og evaluering av gjenfinningsmetodene.

Datainnsamlingen ble gjennomført av søkeroboten som systematisk hentet sangene fra VG-Lista Topp 20 sine lister og komplementære data fra de eksterne APIene, Echonest og MusixMatch. Det er ønskelig å tro at metodene brukt ved innsamling av data var perfekte og feilfrie, men det er dessverre ikke mulig å tilby brukere av systemet en 100% garanti. Innsamling av sangene startet ved webskraping av VG-Lista Topp 20 sine lister. Hvis listene inneholder feilaktige data, slik som at sanger sin artist eller tittel er feilstavet vil det føre til at komplementære data hentet fra de eksterne APIene vil være feilaktige eller ikke-eksisterende.
Hvis vi antar at datainnsamlingen fra VG-Lista Topp 20 er korrekt kan det fremdeles oppstå feilaktige data i datainnsamlingen. Dette er fordi de eksterne APIene også kan returnere feilaktige data fra sine metoder. APIene Echonest og Musixmatch er som nevnt tidligere, valgt på bakgrunn av deres dominerende posisjon i det kommersielle markedet, men det betyr ikke at man skal stole blindt på deres tjenester. Det er derfor mulig at noen av sangtekstene fra Musixmatch og akustiske egenskaper fra Echonest ikke er korrekte. Feilaktige data i ett eller flere av disse stegene i datainnsamlingen vil svekke systemet, ettersom det fører til korrupte inputverdier for algoritmen for akustiske egenskaper og gjenfinningsmetodene, noe som igjen fører til presentasjon av feilaktige data i web-applikasjonen.

Det finnes dessverre ingen effektive metoder for å sikre at datainnsamling for tusenvis av sanger er feilfrie og at de beholder sin integritet. En slik metode vil kun være i stand til å løse problemer man selv identifiserer, men ikke tilpasse seg etter uforutsette problemer. For å oppdage uforutsette problemer kan man benytte seg av de implementerte visualiseringene, ettersom man kan enkelt se objekter som skiller seg ut. Et eksempel på dette er i Figur 6.1 hvor man tydelig kan se at det er noen objekter som skiller seg veldig ut fra resten av mengden. Slike observasjoner kan videre analyseres for å finne ut om det er korrupte eller reelle data.



Figur 6.1: Visualisering av feilaktig data

Figur 6.1 viser sangen "Your Rock My World" av Michael Jackson i det implementerte spredningsplottet. Jeg har iallfall aldri hørt versjonen av "You Rock My World" som er på ca. 14 minutter. Å oppdage slike problemer er enkelt og effektivt ved bruk av visualiseringer. Dette fører dessverre til at visualiseringene blir et "tveegget sverd", da de objektene som skiller seg ut kan enten være korrekte og vise interessante mønstre eller feilaktige og "lyve" til brukeren. Brukere er derfor nødt til å bruke sin egen oppfatning av dataene sammen med visualiseringen for å vurdere resultatet.

Evaluering av gjenfinningsmetodene er basert på målet average precision@5 over 50 test-spørringer og gir det Baeza-Yates & Ribeiro-Neto(2011, s. 140) kaller en tidlig vurdering av hvilke av gjenfinningsmetodene LSI og TF-IDF som er å foretrekke. Det er på bakgrunn av denne "tidlige vurderingen" og analysering av noen test-spørringer hvor LSI presterte dårlig som gjorde at valget av gjenfinningsmetoden for systemet ble TF-IDF.
Valg av hvilke returnerte dokument fra gjenfinningsmetodene som var relevante for test-spørringen ble utført av meg selv og påvirker derfor resultatene til en viss grad. Dette er en nødvendig faktor i evaluering av informasjonssystemer, ettersom man trenger i de fleste tilfeller "menneskelige-eksperter" til å finne de relevante dokumentene for en gitt spørring. Slik at man kan vurdere metodene sin precision for spørringene, osv. De relevante dokumentene for en spørring er derfor

påvirket av min subjektive mening og andre vil mest sannsynlig ha forskjellige meninger om hvilke dokumenter som er relevante. En måte dette kunne blitt forbedret på er ved å la en eller flere musikk-eksperter evaluere om dokumentene(sangtekstene) er relevante eller ikke, slik at det ville vært mulig å ha flere enn 50 test-spørringer.

Manning & Raghavan (2009, s. 140) mener at 50 test-spørringer er nok til å få et godt innblikk i hvordan de forskjellige gjenfinningsmetodene presterer. Jeg mener at dette er litt få, sett opp mot at systemet har 2, 745 sanger med sangtekst. Sett i retrospekt skulle jeg ønsket å hatt tid til å evaluere gjenfinningsmetodene grundigere, for å få et enda sikrere svar på hvilken metode som presterte best.

6.3 Fremtidig arbeid

Målet med denne avhandlingen har vært å skape en fungerende prototype av et system som støtter gjenfinning og visualisering av relasjoner i norsk popmusikk. Systemet har vært i stand til å oppfylle de mest grunnleggende kravene, og vist at konseptet er gjennomførbart. Videre er det flere interessante måter systemet kan utvides, for å øke sin verdi hos brukere, og til forskere.

En interessant utvidelse kunne vært at brukere selv kan bestemme kriterier for sammenligning av akustiske data. For eksempel at brukeren ville bestemt hvilke akustiske egenskaper som ville blitt brukt i algoritmen, slik at brukeren ville funnet like sanger basert på sine egne kriterier. Dette kunne blitt implementert ved å ha et applikasjonsprogrammeringsgrensesnitt(API) med metoder som ville tatt hensyn til kriteriene og spurt back-enden om å produsere resultatene fra databasen.

En annen ting er at det er antatt at musikkinteresserte personer, historikere, musikkjournalister, produsenter og låtskrivere vil finne systemet nyttig. Systemets web-applikasjon er derimot bare testet på musikkinteresserte personer i usability-testingen. Det hadde derfor vært interessant å teste systemet på de forskjellige typer brukere for å se om de ville tatt det i bruk slik som antatt. Dette kunne blitt gjennomført ved å sende ut en elektronisk spørreundersøkelse som de skulle ha besvart etter de hadde prøvd web-applikasjonen.

Implementasjon av metoder fra fagområdet musikk informasjonsgjenfinning ville vært en stor, men spennende utvidelse. Fagområdet omhandler hovedsakelig gjenfinning av musikk informasjon ved å hente ut informasjon fra lydfiler. Det implementerte systemet baserer seg allerede på informasjon fra lydfiler, men disse akustiske egenskapene er innhentet via det eksterne APIet Echonest og er derfor allerede prosessert til de egenskapene de mener er interessante.

Ved å ha tilgang til lydfilene til sangene kunne slik data blitt produsert selv ved å analysere filene. Dette kunne blitt gjort ved bruk av Marsyas[20], som er et åpen kildekode rammeverk for lydprosessering med spesiell vekt på musikk informasjonsgjenfinning(Music Information Retrieval) applikasjoner. Data generert fra lydsporene kunne da blitt brukt i musikk informasjonsgjenfinningsmetoder som kunne gjort det mulig for systemet å finne sanger basert på små lydspor(fingerprinting), gjenkjenne og klassifisert tekst fra lydspor(speech recogniztion), gjenkjenne personen som snakker i et gitt lydspor(speaker identification) og finne lydspor som matcher en tekst spørring (Baeza-Yates & Ribeiro-Neto, 2011, s. 598). Denne ideen ble vurdert i starten av prosjektet, men ble vurdert til å være for avansert og tidskrevende å implementere sammen med det som er blitt gjort.

6.4 Konklusjon

Mitt mål i denne avhandlingen var å svare på spørsmålet om hvordan man kan støtte gjenfinning og visualisering av relasjoner i norsk popmusikk. Gjennom avhandlingen har implementasjonen og evalueringen av et system som støtter dette blitt beskrevet og funnene har resultert i positive resultater for forskningsspørsmålet.
I det endelige systemet har jeg implementert en gjenfinningsmetode som oppdager relasjoner mellom sanger basert på sangtekster, og en algoritme som oppdager relasjoner mellom sanger basert på akustiske egenskaper. Disse relasjonene og andre data i norsk popmusikk er blitt visualisert til brukere på en brukervennlig måte som gir mening.

Det er flere ting listet i fremtidig arbeid som må implementeres for at et slikt system kan sies å være komplett, men gjennom utviklingen av prototypen er det vist at konseptet er gjennomførbart. Selv om det er gjort noen avgrensninger i evalueringen, som gjør at jeg ikke kan konkludere med noe, så føler jeg meg trygg på at prototypen har vist et mulig system som støtter gjenfinning og visualisering av norsk popmusikk.

[20] https://github.com/marsyas/marsyas

Referanseliste

Baeza-Yates, R., & Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology behind Search*. *Information Retrieval* (Vol. 82). Retrieved from http://www.amazon.com/Modern-Information-Retrieval-Concepts-Technology/dp/0321416910

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, *4*(3), 114–123. http://doi.org/66.39.39.113

Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*. http://doi.org/10.1137/1037127

Bevan, N. (1995). Measuring usability as quality of use. *Software Quality Journal*, *4*(2), 115–130. http://doi.org/10.1007/BF00402715

Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, *189*, 194. http://doi.org/10.1002/hbm.20701

Brooke, J. (1996). Usability Evaluation In Industry. In *Usability evaluation in industry* (pp. 189–194). Retrieved from http://books.google.co.nz/books?hl=en&lr=&id=IfUsRmzAqvEC&oi=fnd&pg=PA189&dq=Brooke,+J.+(1996).+SUS:+A+?quick+and+dirty?+usability+scale.&ots=G8rtzaok2g&sig=78YYKo8R_qkNNoUifjgsV3QgAxQ#v=onepage&q&f=false

Cockburn, A. (2001). *Writing effective use cases*. *ACM SIGSOFT Software Engineering Notes* (Vol. 26). http://doi.org/10.1145/505894.505918

Dalhuijsen, L. (2011). *WITH · MusicalNodes*. *Most*. Retrieved from http://studiowith.nl/projects/musicalnodes

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*, 391–407. http://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Dumas, J. S., & Redish, J. C. (1999). *A practical guide to usability testing*. *Star* (Vol. Rev. ed.). Retrieved from http://www.amazon.com/Practical-Guide-Usability-Testing/dp/1841500208

Ellis, S., & Engelhardt, T. (2011). Visualizing a Hit. Retrieved April 17, 2015, from https://sites.google.com/site/visualizingahit/

Friedman, V. (2008). Data Visualization and Infographics. Retrieved from http://www.smashingmagazine.com/2008/01/14/monday-inspiration-data-visualization-and-infographics/

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105. http://doi.org/10.2307/25148625

Hilliges, O., Holzer, P., Klüber, R., & Butz, A. (2006). AudioRadar: A metaphorical visualization for the navigation of large music collectionsNo Title. Vancouver, Canada. Retrieved from http://link.springer.com/chapter/10.1007%2F11795018_8#page-1

Jones, K. S. (1972). A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*. http://doi.org/10.1108/eb026526

Kobourov, S. (2012). Spring Embedders and Force Directed Graph Drawing Algorithms. *arXiv Preprint arXiv:1201.3011*, 1–23. Retrieved from http://arxiv.org/abs/1201.3011

Kurniawan, T. (2009). Infographics & Data Visualisation. Retrieved from http://www.slideshare.net/trisnadi/infographics-data-visualisation

Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. *Analysis*.

Lewis, J. R., & Sauro, J. (2009). The factor structure of the system usability scale. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5619 LNCS, pp. 94–103). http://doi.org/10.1007/978-3-642-02806-9_12

Luhn, H. P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*. http://doi.org/10.1147/rd.14.0309

Manning, C. D., & Raghavan, P. (2009). An Introduction to Information Retrieval. (A. C.-B. E. Salas, Ed.)*Online*. Cambridge University Press. http://doi.org/10.1109/LPT.2009.2020494

Murray, S. (2013). *Interactive Data Visualization for the Web* (2nd ed.). O'Reilly Media.

Nielsen, J. (1993). *Usability Engineering*. *Usability Engineering* (Vol. 44). http://doi.org/10.1145/1508044.1508050

Nielsen, J. (1994). Estimating the number of subjects needed for a thinking aloud test. http://doi.org/10.1006/ijhc.1994.1065

Pampalk, E. (2003). Islands of music analysis, organization, and visualization of music archives. *OGAI Journal (Oesterreichische Gesellschaft Fuer Artificial Intelligence)*, *22*(4), 20–23. http://doi.org/10.1.1.4.5107

Rosario, B. (2000). *Latent Semantic Indexing: An overview*. Retrieved from http://www.cse.msu.edu/~cse960/Papers/LSI/LSI.pdf

Rubin, J. (1994). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. New York, NY: John Wiley & Sons, Inc.

Salton, G., & Yang, C. S. (1973). On the specification of term values in automatic indexing. *Journal of Documentation*. http://doi.org/10.1108/eb026562

Sauro, J. (2011a). A practical guide to the System Usability Scale: Background, benchmarks.

Sauro, J. (2011b). Measuring Usability With The System Usability Scale (SUS).

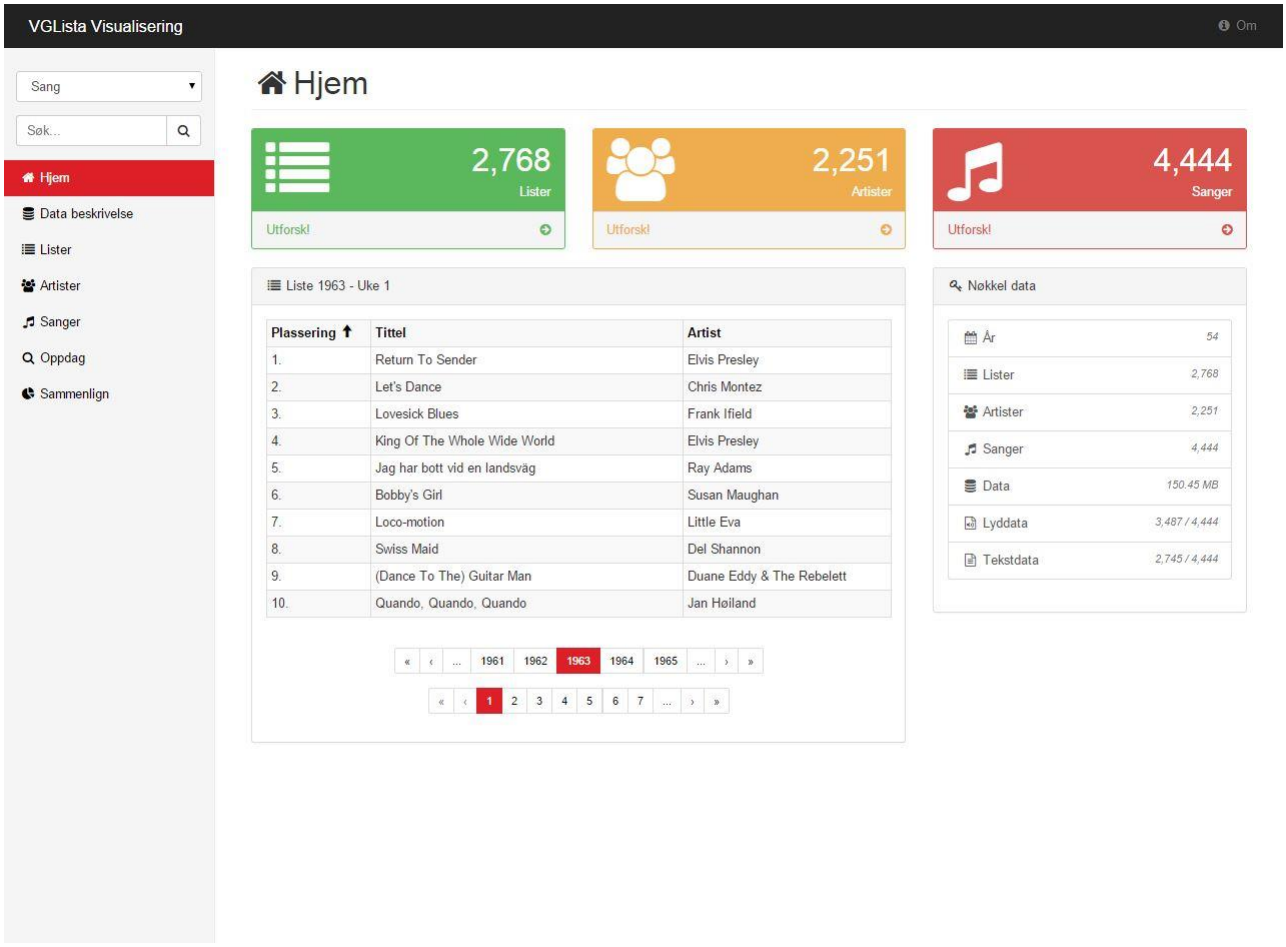
Silva, J. (2014). End-to-end javascript with the mean stack. Retrieved from http://joaopsilva.github.io/talks/End-to-End-JavaScript-with-the-MEAN-Stack/#/2

Torrens, M., Hertzog, P., & Arcos, J. (2004). Visualizing and exploring personal music libraries. In *Proc. 5th International Conference on Music Information Retrieval (ISMIR '04)* (pp. 421–424). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.9216&rep=rep1&type=pdf

Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. *Technometrics* (Vol. 2nd). http://doi.org/10.1198/tech.2002.s78

Tullis, T. S., & Stetson, J. N. (2004). A Comparison of Questionnaires for Assessing Website Usability. *Usability Professional Association Conference*, 1–12. Retrieved from http://home.comcast.net/~tomtullis/publications/UPA2004TullisStetson.pdf

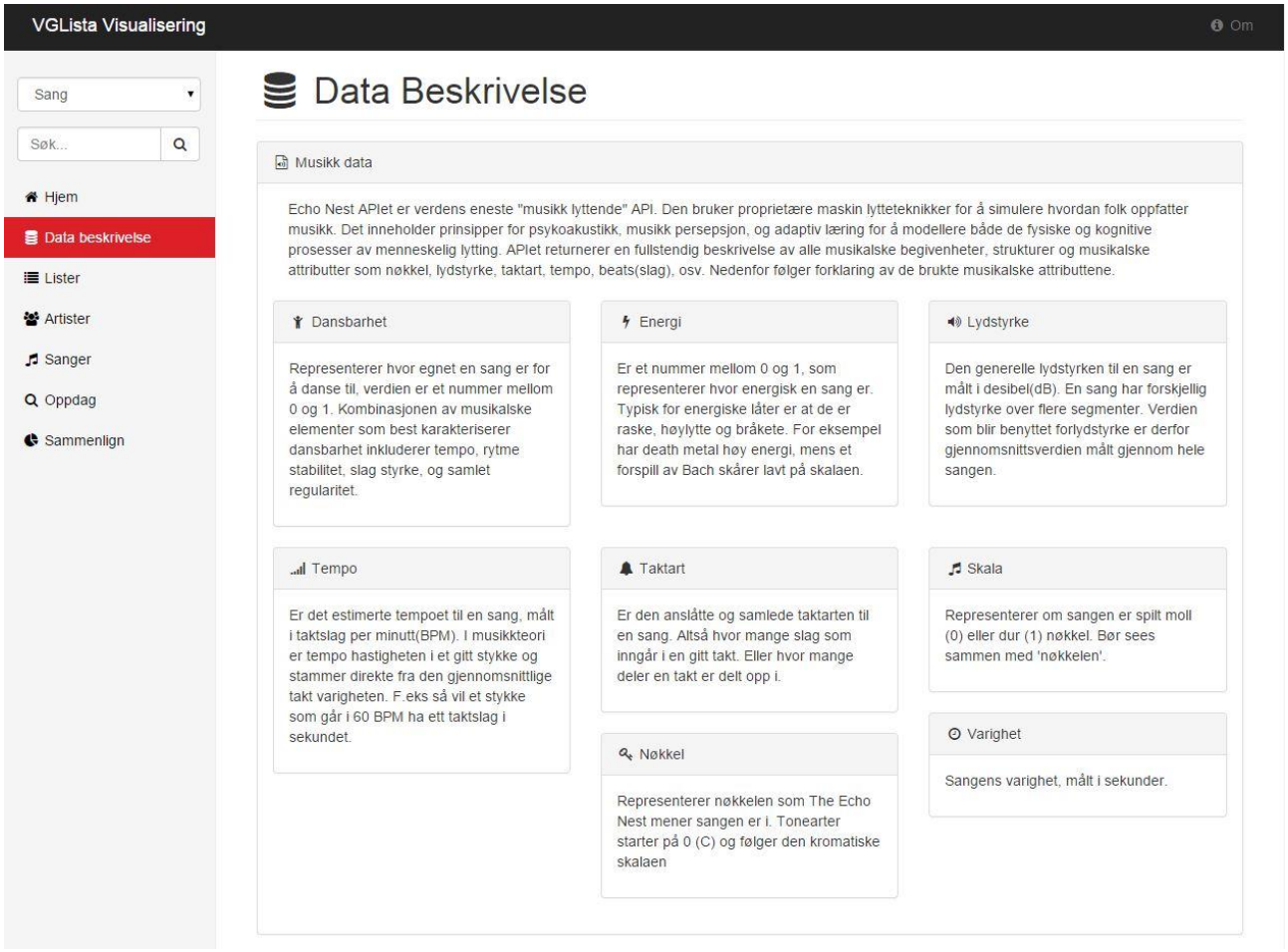
Appendiks A Startside



Web-applikasjonen sin startside inneholder nøkkeldata om prosjektet, slik som antall lister, artister og sanger. Dette er for at brukere skal kunne få en rask introduksjon til den store mengden av data de kan oppdage og utforske. Videre er det implementert en dynamisk liste som gjør det mulig å bla seg gjennom alle VG-Lista Topp 20 listene fra 1960 og frem til 2014.

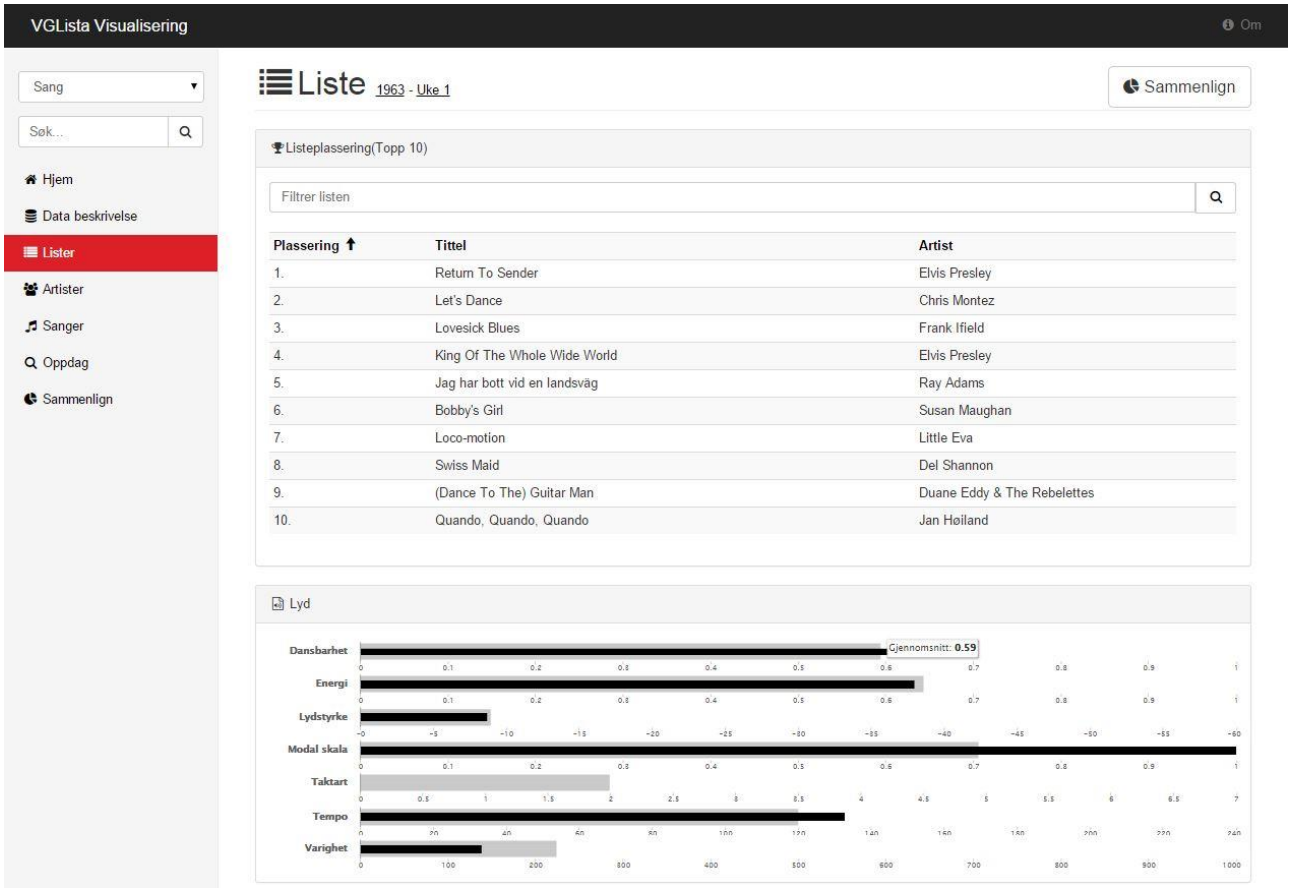
Langs venstre siden er hovedmenyen, som inneholder lenker til alle de forskjellige sidene. Det er også implementert en søkemotor, som lar brukere enkelt finne frem til forskjellige artister, lister eller sanger avhengig av hva de ønsker å søke etter.

Appendiks B Databeskrivelse



Databeskrivelse-siden presenterer de forskjellige typer data som brukere kan komme over i web-applikasjonen. Data som blir beskrevet er de forskjellige akustiske egenskapene, som dansbarhet, energi, lydstyrke, tempo, osv. Det er også forklart litt om sangtekstene og prosessering av de, slik som at ordene sin frekvens er talt og at stoppord er fjernet. Dette er for at brukere skal kunne forstå de forskjellige dataene.

Appendiks C Liste



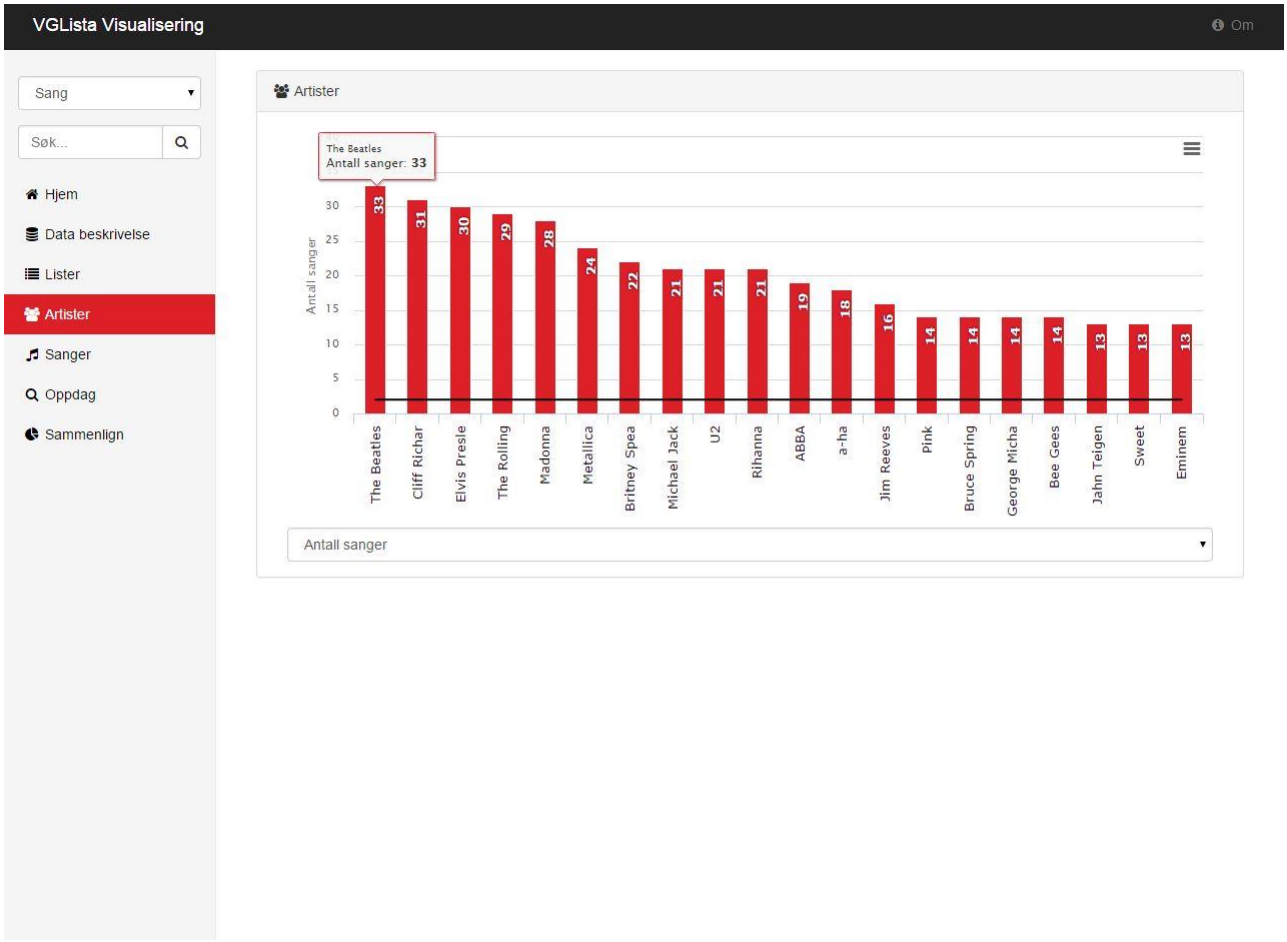
Liste-siden inneholder data om en gitt liste. Dette er selve listen med sangene og deres tittel, artist og plassering. Listen kan søkes i og sorteres ved klikk på tabell-overskriftene, slik at det er enkelt å filtrere listen. Sangene i listen fungerer som lenker, slik at det er enkelt å navigere seg frem til mer data om en sang.

De akustiske egenskapene til listen presenteres i kulegrafer, slik at det er mulig å se sammenhengen mellom listen sine verdier og de andre listene. Den sorte stolpen presenterer verdien for den gitte listen og den grå stolpen er et komplementært mål som presenterer gjennomsnittet av verdien i alle lister.

Alle ordene som er brukt i listen er også presentert i form av en ordsky, se seksjon 4.10.1. Ordskyen presenterer ordene ved å fremheve de mest brukte ordene.

Det er også en ”sammenlign”-knapp øverst i høyre hjørne, slik at det er mulig å raskt velge at man vil sammenligne listen med andre.

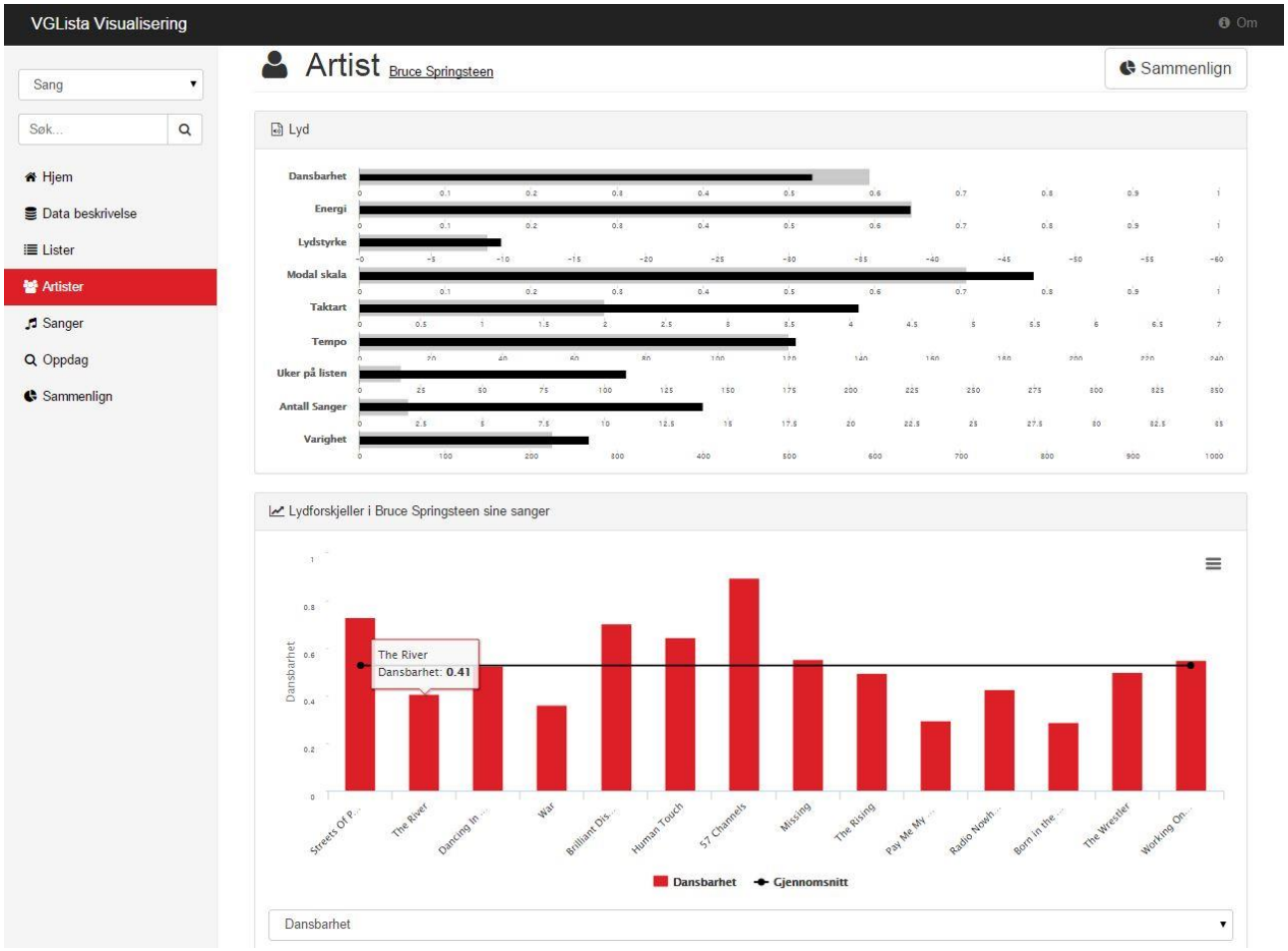
Appendiks D Artister



Artister-siden inneholder et stolpediagram med 20 artister som har høyest verdi for en gitt akustisk egenskap brukeren har valgt å se på. Brukere velger ønsket egenskap ifra nedtrekkslisten lokalisert under diagrammet.

Den sortelinjen i stolpediagrammet representerer gjennomsnittsverdien for den gitte egenskapen for alle artister, slik at brukere kan få et forhold til verdien i de forskjellige stolpene. Tekst-etikettene under stolpene fungerer som lenker, slik at man enkelt kan besøke artist-siden for å få mer informasjon. Nedtrekkslisten inneholder de vanlige akustiske egenskapene, men også antall sanger og antall ganger listet. Disse to egenskapene er spennende å se på hvis man ønsker å finne ut av hvilke artister som har vært mest populære i norsk popmusikk.

Appendiks E Artist



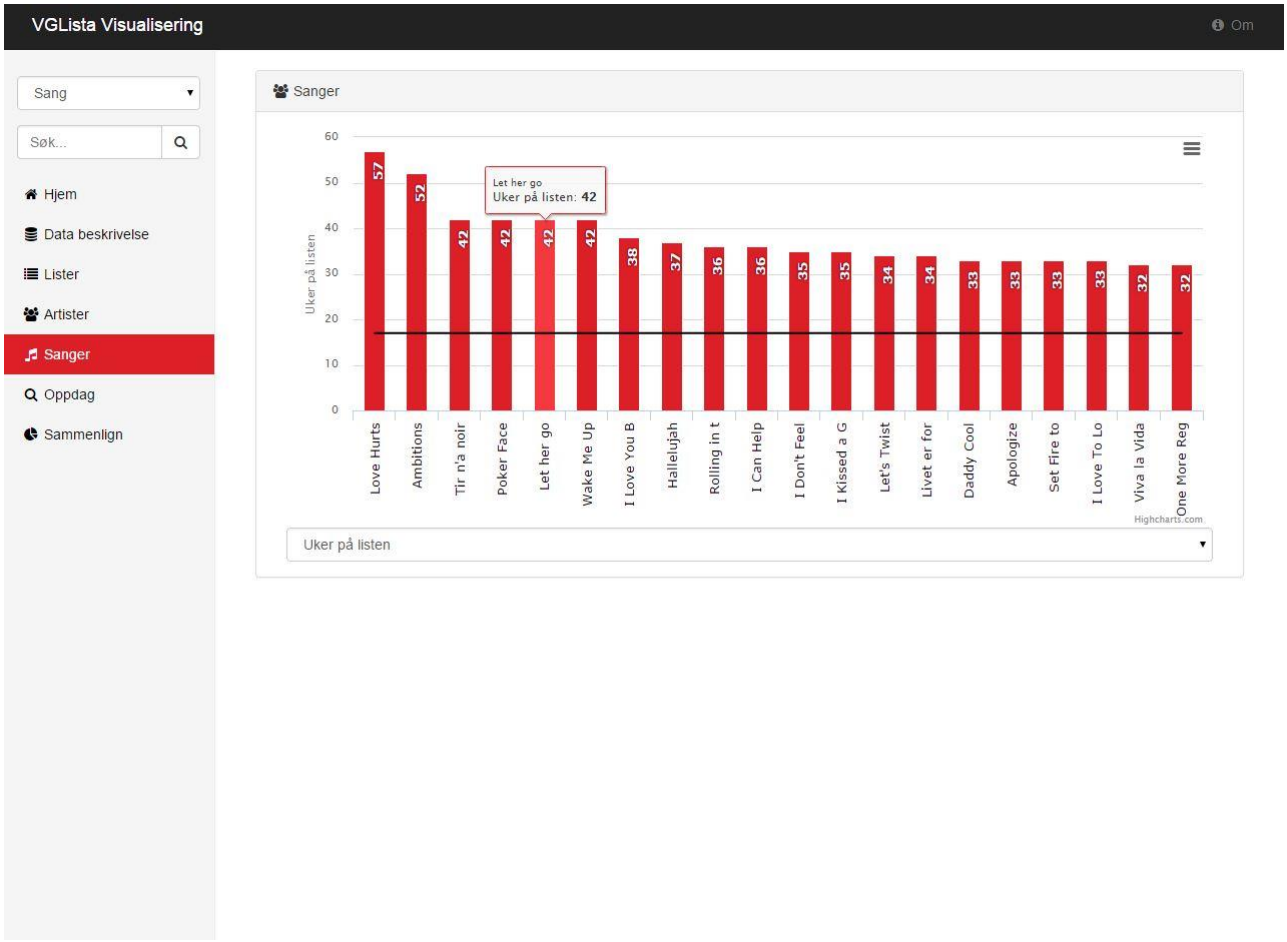
Siden inneholder data om en gitt artist og er veldig lik som liste-siden, presentert i appendiks 0. For å presentere de akustiske egenskapene til artisten blir kulegrafen brukt.

Presentasjon av artisten sin lyrikk er implementert ved å bruke ordskyen, som er presentert i seksjon 4.10.1 Videre er det en linjegraf som viser plasseringene til artisten opp gjennom årene på VG-Lista Topp 20 listene. Grafen gir en god oversikt over artisten sine plasseringer.

For å kunne presentere endringer i de akustiske egenskapene til artisten sine sanger, er det implementert et stolpediagram hvor man kan endre akustisk egenskap fra nedtrekkslisten. Stolpediagrammet gjør det mulig for brukere å sammenligne de forskjellige akustiske egenskapene til en artist sine sanger. Det er for eksempel enkelt å se forskjellen i tempo for de forskjellige sangene.

I bunn av siden er det implementert en liste over artisten sine sanger med data om hvor mange uker sangen har vært listet, sammen med dens beste plassering. Sangene i listen fungerer også som lenker, slik at man enkelt kan navigere til en sang sin side for å få mer informasjon.

Appendiks F Sanger



Sanger-siden er så å si helt lik artister-siden, se appendiks 0. Den eneste forskjellen er at data presentert er sanger og ikke artister. Det at de to sidene er like i design er gjort på bakgrunn av at applikasjonen skal være konsistent, noe som fører til en bedre brukervennlighet. Siden inneholder kun et stolpediagram, som presenterer de 20 sangene med høyest verdi for en gitt egenskap brukeren har valgt å se på. Brukere velger ønsket egenskap ifra nedtrekkslisten lokalisert under diagrammet.

Appendiks G Samtykkeskjema.

Jeg samtykker i å delta i datainnsamling til oppgave I INFO390, utført av Anders Langseth.

Jeg forstår at deltakelse i denne usability-testen er frivillig og at testen kan avbrytes når det matte passe seg, for eksempel hvis bekymringer eller ubehag oppstår.

Jeg tillater at min utførelse av usability-testen vil bli dokumentert og brukt som data i masteroppgaven til Anders Langseth. De data som brukes vil ikke stå i mitt navn og vil være anonymisert i selve avhandlingen.

Vennligst skriv under nedenfor for å indikere at du har lest og forstått informasjonen på dette skjemaet og at eventuelle spørsmål du matte ha om økten har blitt besvart.

Dato: _____

Blokkbokstav, ditt navn: _____

Underskrift, ditt navn: _____

Tusen takk!

Jeg setter pris på din deltakelse,
Anders Langseth.

Appendiks H Scenarier.

1. Du jobber som musikkjournalist og er interessert i utviklingen av lydstyrke i norsk popmusikk over de siste 50 årene.
2. Du er interessert i å se på forskjellene mellom dine to favorittsanger, Britney Spears - Womanizer og The Beatles - Yesterday.
3. Du er uenig i en venn sin påståelse om at den mest populære pop-artisten i Norge de siste 50 årene er Madonna og ønsker å finne svar på denne påståelsen.
4. Du er en energisk person som liker nyere sanger(etter år 2000) med høyt tempo og masse energi. Du ønsker å oppdage flere slike sanger.
5. Du har nylig lest en spennende artikkel om en av 2000-tallets største rappere, 50 Cent, hvor han sine tekster blir fremstilt som fulle av skjellsord og rasistiske ord. Du deler ikke mening med artikkelen og ønsker å finne ut av dette.
6. Du har hørt på favorittartisten din, Jim Reeves, i mange år, men det er få personer du møter som liker musikken hans. Du undrer deg selv om artisten Jim Reeves er listet mange ganger i Norsk musikk sin historie.
7. Du er usikker på når Michael Jackson sin sang Thriller ble utgitt og ønsker å finne ut av dette.
8. Du har nylig åpnet øynene for artisten Queen og ønsker å finne flere sanger som denne gruppen har utgitt.
9. Favorittsangen din er Firestone av Kygo og du ønsker å finne flere like sanger.
10. Du ønsker å finne ut hvilke sanger som var listet på VG-Lista Topp 20 i starten av året 2014.

Appendiks I SUS-spørreundersøkelsen.

1. Jeg tror jeg ønsker å bruke dette systemet ofte.
2. Jeg fant systemet unødig komplisert.
3. Jeg syntes systemet var enkelt å bruke.
4. Jeg tror jeg vil trenge støtte fra en teknisk person for å kunne bruke systemet.
5. Jeg fant de ulike funksjonene i dette systemet godt integrert.
6. Jeg synes det var for mye inkonsistens i dette systemet.
7. Jeg forestiller meg at de fleste vil lære å bruke dette systemet svært raskt.
8. Jeg fant systemet veldig tungvint å bruke.
9. Jeg følte meg veldig trygg ved bruk av systemet.
10. Jeg trengte å lære en masse ting før jeg kunne komme i gang med dette systemet.

Hvert spørsmål ble besvart med likert-skalaen:

Helt Uenig Helt Enig

1	2	3	4	5