Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Utvikling av kategoriseringsverktøy for PDF-notesett
Kravdokumentasjon

Versjon 1.2

Dokumentet er basert på Kravdokumentasjon utarbeidet ved NTNU. Revisjon og tilpasninger til bruk ved IDER, DATA-INF utført av Carsten Gunnar Helgesen, Svein-Ivar Lillehaug og Per Christian Engdal. Dokumentet finnes også i engelsk utgave.

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
27/02/22	1.0	Første versjon av kravdokument	Johan M. Engevik
06/05/22	1.1	Andre versjon av kravdokument	Johan M. Engevik
22/05/22	1.2	Tredje versjon av kravdokument	Johan M. Engevik

INNHOLDSFORTEGNELSE

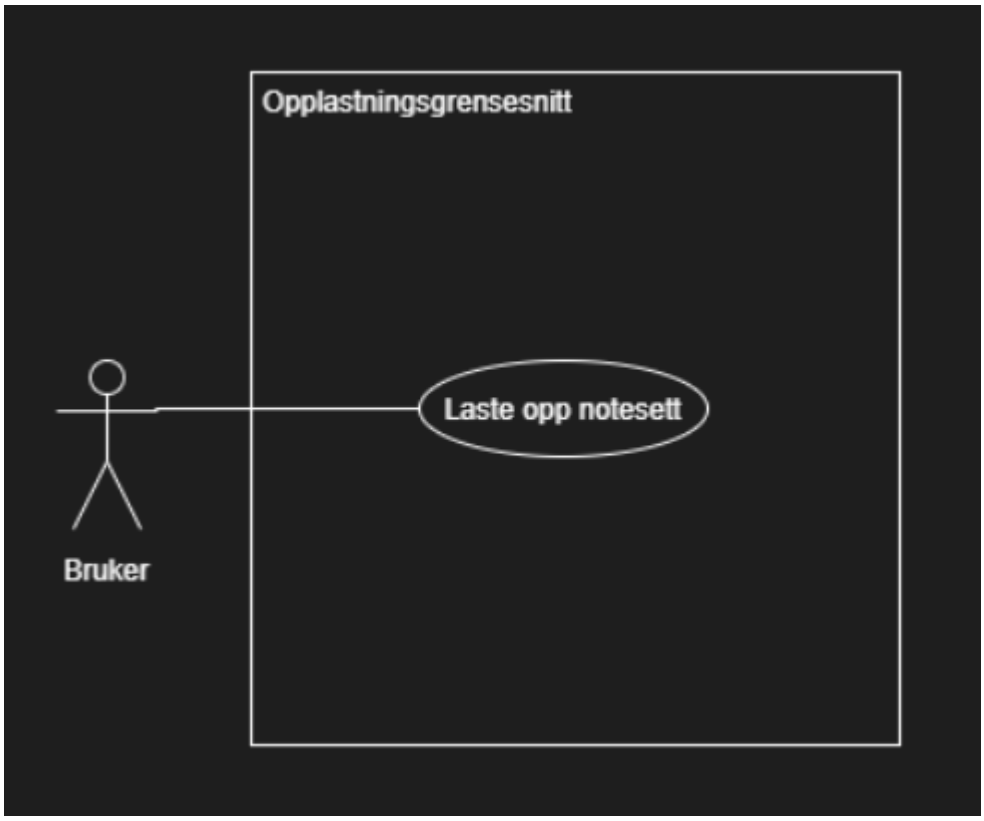
1 INNLEDNING

Dette dokumentet er til for å gi et grunnlag for en løsning som tilfredsstiller kravene fra oppdragsgiver. Etter hvert som utviklingen kommer lengre, vil informasjonen her oppdateres. I denne førsteversjonen av dokumentet beskrives i utgangspunktet en ide om hvordan problemet kan løses. Denne ideen beskrives ved hjelp av brukstilfelle diagram og domene modell. I tillegg nevner vi litt om den første prototypen for notetolkning.

2 Funksjonalitet

Systemet som vi skal utvikle har i utgangspunktet bare et brukstilfelle, og det er når en bruker ønsker å laste opp et nytt notesett til notearkivet. Det skal i utgangspunktet ikke gjøres noe med grensesnittet som allerede er i bruk.

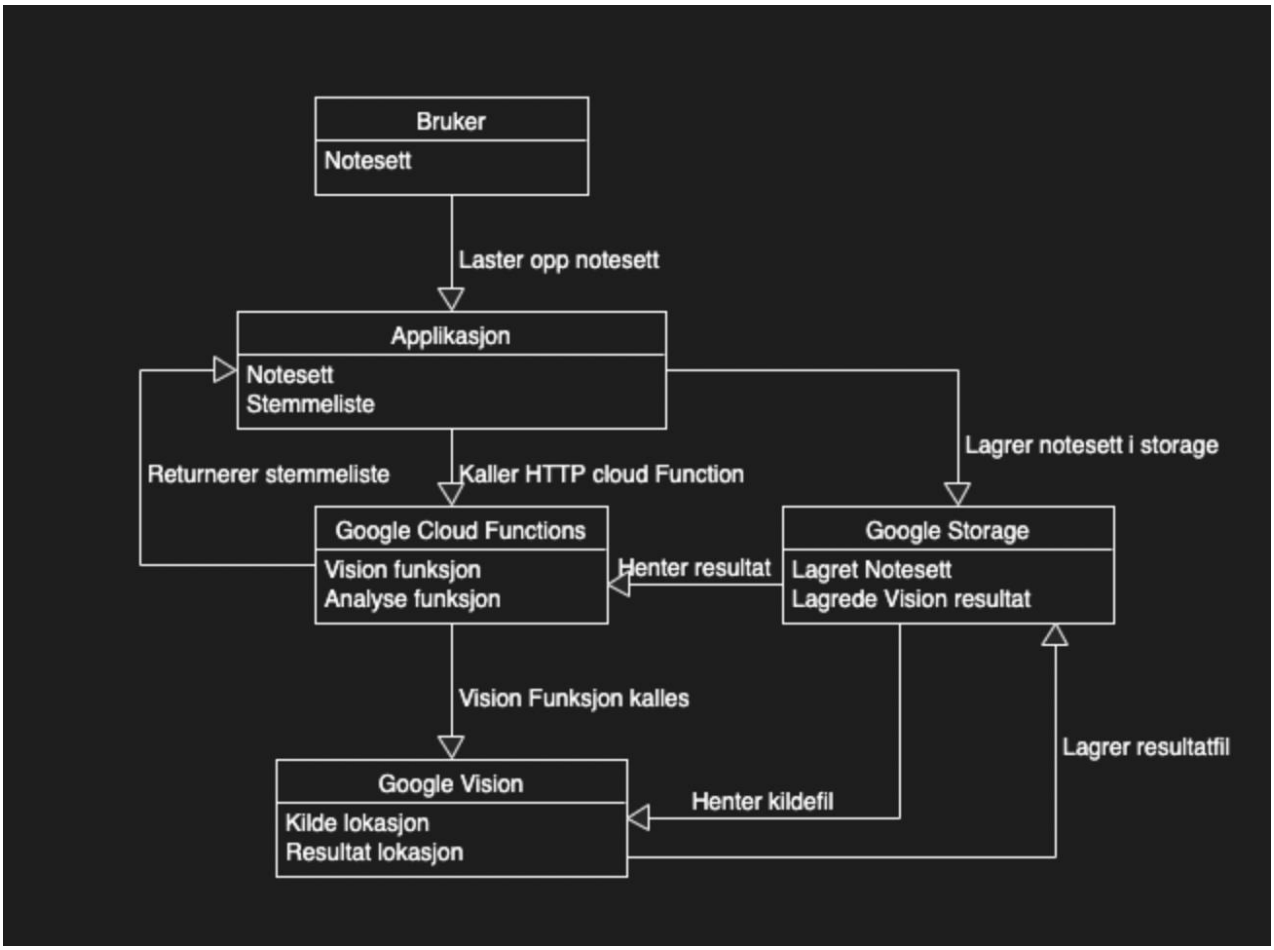
Bruker laster opp ett notesett og systemet vil da automatisk forsøke å tilordne hver side til en stemme ved hjelp av tekst gjenkjenning. Dersom systemet ikke klarer å tilordne en stemme, så må bruker manuelt tilordne siden en stemme



Figur 1- Brukstilfelle diagram

Navn:	Laste opp nytt notesett.
Aktører:	Noteforvaltere i korps (bruker)
Hensikt/Målsetting:	Laste opp et nytt notesett i pdf filformat, og splitte det opp i de individuelle stemmene automatisk.
Normalflyt:	1. Bruker velger notesettet som skal lastes opp, enten ved filutforsker eller ved drag/drop funksjon. 2. Bruker klikker på last opp og filen lastes opp til Databasen. 3. Systemet sender så relevante deler av sidene til Google Vision for tekstgjenkjenning og får tilbake en JSON streng med hva som detektert, avgrensningsbokser og konfidensverdier. 4. Systemet bruker utlest informasjon til å bestemme hvilken stemme hver side hører til og viser resultatet til bruker
Alternativ flyt [#1[:	• Dersom konfidensverdien er for lav, eller systemet ikke finner noen match for stemmen, ber den bruker om manuell tilordning for de gjeldende sidene
Alternativ flyt [#2]:	• Dersom bruker ser at en stemme ikke stemmer så kan de manuelt endre hvilken stemme en side tilhører før notesettet lagres.
Unntaksflyt [#1]:	• Dersom filen ikke kan leses, skal det sendes en feilmelding om dette, og bruker bes forsøke å laste opp på nytt.

2.1 DOMENEMODELL



Figur 2 - Domenemodell

I figur 2 vises domenemodellen. Den beskriver prosessen som skal skje når brukeren har lastet opp et notesett. Først lagres notesettet midlertidig i Google Storage og Google Vision kalles gjennom en «Cloud Function» for å tolke notesettet som er lagt inn. Resultatet av denne tolkningen eksporteres tilbake til Google Storage av Google Vision. Deretter ber applikasjonen en ny «Cloud Function» om å analysere disse dataene. Funksjonen må så gjenkjenne stemmen som hører til den enkelte PDF siden, og returnerer deretter en liste over stemmene den fant i rekkefølge.

3 PROTOTYPER

3.1 Første iterasjon

For å teste funksjonaliteten i Google sitt Vision API, bruker vi et script i Python som baserer seg mye på eksempelkoden fra Google sin dokumentasjon [1]. Det går fint å hente ut informasjonen fra de skannede notene, men på større notesett noterer vi at det tar noe lengre tid å produsere resultater.

Det ble også testet litt metoder for sammenligning av tekststrenger for å finne stemmen som er mest lik resultatet som ble lest. I våre tester brukte vi Levenshtein distanse. Levenshtein Distanse går ut på å måle hvor mange enkelt tegn som må forandres, fjernes eller legges til for at en streng skal bli lik en annen [2].

3.2 Andre iterasjon

Vi ønsker å bedre muligheter for å teste algoritmen vår og lager derfor en demoapp på nett som lar en bruker laste opp et PDF notesett og få tilbake en liste over stemmene i rekkefølgen de var lastet opp. For å få dette til er vi nødt til å benytte oss av Google Cloud Functions for å kunne lage implementere funksjoner på tjenersiden. Vi lager en funksjon som kaller Vision APIet, og en annen funksjon som analyserer resultatet fra Vision. Funksjonen som kaller Vision APIet er i all hovedsak basert på eksempelkode fra Google sin dokumentasjon [1]. Resultatet er et JSON objekt med en respons for hver side i PDF filen. Hver respons har et felt kalt fullTextAnnotation som igjen har et felt kalt tekst. I dette feltet er all teksten på siden samlet i en lang streng. Vi splitter strengen på «\n» og sammenligner alle ordene med en liste over stemmer i brass og janitsjar band. Ordet som har lavest Levenshtein distanse med en stemme, blir satt som stemmen til siden.

3.3 Tredje iterasjon

Vi opplever at å søke gjennom tekstfeltet fra start til slutt ikke alltid finner rett stemme. Eksempelvis får vi feil stemmer dersom tittelen på stykket inneholder et instrumentnavn som eksempelvis «Cornet Solo». For å løse dette problemet må vi ta i bruk en annen del av resultatobjektet.
JSON objektene som Vision generer består av et Kontekst felt som inneholder informasjon om hvor filen som ble analysert er lagret, og en liste med responser. Hver PDF side som er analysert blir til en respons i den listen. Hver respons har et nytt felt kalt fullTextAnnotation som igjen har to underfelter. Text som inneholder all teksten på siden i en lang streng, og blocks som inneholder en liste av tekstblokker på siden. Hver tekstblokk har igjen en liste over paragrafer, hver paragraf har en liste over ord, og hvert ord har en liste over symboler. Hvert symbol har igjen et tekstfelt som inneholder det faktiske tegnet.

3.4 Fjerde iterasjon

Vi opplever at Levenshtein distanse ikke gir god nok sammenligning. Sekvensielt like bokstaver får ikke noen preferanse. Vi bytter derfor stengsammenligningsalgoritme til Jaro-Winkler-Metric [3]. Denne algoritmen gir oss mer nøyaktig klassifisering av stemmer. Vi har også implementert funksjonalitet som gjør at søkealgoritmen prøver å finne de bitene med tekst som er plassert omtrent der hvor stemmen pleier å stå skrevet.

4 REFERANSER

[1] Google, «Detect text in files (PDF/TIFF),» 2022. [Internett]. Available: https://cloud.google.com/vision/docs/pdf. [Funnet 27 Februar 2022].

[2] M. Gilleland, «Levenshtein Distance, on Three Flavors,» [Internett]. Available: https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm. [Funnet 27 Februar 2022].

[3] W. E. Winkler, «String Comparator Metrics and Enhanced Decision Rules in the Fellefi-Sunter Model of Record Linkage,» U.S. Bureau of the Census Stat. Research Div., Washington DC, 1990.