NTNU
Det skapende universitet

Klassifisering av sykepleiejournalen

Kan kunnskap om sykepleiedokumenter
forbedre gjenkjenning av hendelser knyttet til
sentralvenekateterisering?

Haldor Husby

Helseinformatikk
Innlevert: juli 2014
Hovedveileder: Øystein Nytrø, IDI
Medveileder: Laura Slaughter, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Forord

Denne masteroppgave er avslutningen på min videreutdanning ved NTNU innenfor helseinformatikk. Oppgaven er gitt og skrevet for forskningsprosjektet Evicare og NTNU. Veileder for prosjektet har vært Øystein Nytrø, men jeg har også fått god bistand fra Thomas Brox Røst og Laura Slaughter.

Jeg vil takke min familie for de utallige timer jeg har jobbet med oppgaven. Jeg vil også takke mine medstudenter, kolleger og tidligere kollegaer for all bistand de har gitt meg. Spesielt vil jeg takke Christine Tvedt for den medisinske forklaringen og ikke minst hennes jobb med annoteringen av materialet.

Sammendrag

Det meste av dokumentasjon om behandlingen en pasient får ved et sykehus er i form av tekstlige dokumenter. Sentralt venekateter (SVK) er et viktig medisinsk utstyr som gir tilgang til blodbanen. Det er stor risiko for pasienten å ha SVK liggende lenge. I tillegg er dokumentasjon på hvilke pasienter som har SVK vanskelig å finne ved å bruke den strukturerte delen av EPJ.

Forskningsprosjektet Evicare har hentet 45614 dokumenter hvor en del av disse dokumenter er sykepleierdokumentasjon. Prosjektet har lagd en ontologi for SVK, og utarbeidet en gullstandard for materialet. Gjennom gullstandarden er det 1356 dokumenter som har blitt merket med stell av SVK(CareCVC). Etter en manuell gjennomgang av disse dokumenter har vi funnet ut at 1141 filer innehar en eller flere emneoverskrifter av malen sykepleiere dokumenterer etter. Når vi ser på fordelingen av hvor i malen stell av CVK finnes, viser det seg at Del6 (Hud/vev/sår) i malen skiller seg ut, med 788 dokumenter.

Prosjektet har identifisert 8219 sykepleiedokumenter som har data i Del6 (Hud/vev/sår). Dette danner korpus for studien. Vi har, på bakgrunn av disse dokumenter, lagd to datasett: et datasett som innehar all tekst innenfor malen, og ett datasett som kun inneholder data innenfor Del6 (Hud/vev/sår). For begge datasett har vi i tillegg brukt metoder for forbehandling av tekst, hentet fra språkteknologi, for å redusere variasjon i tekstmaterialet. Blant flere mulige har vi valgt å ekspandere forkortelser og å stemme, det vil si å redusere ord til en standardisert rotform. Vi har da totalt har tre datasett, uhildet dokumenter, dokumenter hvor en del av forkortelsene som finnes er ekspandert og til slutt dokumenter hvor stemming er foretatt etter at en del forkortelser er ekspandert. I tillegg har vi skilt dokumentene etter kjennskap til stell av SVK ved å bruke gullstandarden.

Videre er hvert datasett delt opp i to deler, trening og test, med fordelingsnøkkelen 80 % og 20 % av filene som ble identifisert som sykepleiedokumenter. Læringsalgoritmen Naive Bayes er brukt for å klassifisere dokumentene. Resultatet viser at det å bruke hele dokumenter, uten endringer som forkortelser og stemming, innenfor malen gir precision på 51,5 %, recall på 22,3 % og F_1-score på 31,1 %. Analysene viser at ekspandering av forkortelser isolert sett ikke forbedrer resultatet. Ved å foreta ekspandering av forkortelser og så stemming på delen som kun har data innenfor Del6, gir de beste resultatene for studien med precision 65,7 %, recall på 59,9 % og F_1-score på 62,7 %. Resultatene bekrefter at klassifiseringen av stell av CVK blir bedre ved å se på deler av sykepleierdokumentasjonen i motsetning til alt innhold innenfor malen. Resultatene blir også bedre ved å foreta preprosessering som ekspandering av forkortelser og stemming. Vi har også funnet ut at hendelsen stell av SVK forekommer mest innenfor Del6 av sykepleiemalen.

Abstract

Classification of nursing journal

Most of the documentation about the treatment a patient gets when hospitalized is in the form of textual documents. Central venous catheter (CVC) is an important medical device that provides access to the bloodstream. There is an increased risk for the patient to have CVC for a longer period of time. In addition, evidence of which patients have CVC is difficult to find using the structured part of the EMR.

The research project Evicare has extracted 45,614 documents from the EMR, where some of these documents are nursing documentation. The project has created an ontology for CVC, and prepared a gold standard for parts of the material. Through the gold standard 1356 documents that have been tagged with the CareCVC. After a manual review of these documents, we found that 1141 file holds one or more subject headings of the template document for nurses. When we look at the distribution within the template, care of CVC is found in 788 documents within part 6 (skin / tissue / wound) in the template.

The project has identified 8,219 nursing documents that have data in part 6 (skin / tissue / wound) within the template. This forms the corpus of this study. We have, on the basis of these documents, created two data sets: a data set that holds all text within the template, and one data set that only contains data within part 6 (skin / tissue / wound). For both data sets, we have also used methods for preprocessing of text, taken from Natural Language processing, to reduce variation in the text material. Among several possible we have decided to expand abbreviations and stemming, that is, to reduce words to a standardized root word. We then have a total of three sets of data, unchanged documents, documents that have some of the abbreviations that exists expanded and documents where stemming is done after some abbreviations are expanded. In addition, we split the documents by knowledge of care of CVC using the gold standard.

Furthermore, each data set divided into two parts, training and testing, with the distribution key 80% and 20% of the files that were identified as nursing documents. Naive Bayes learning algorithm is used to classify documents. The results show that using the whole document, without modification as abbreviations and stemming, within the template provides precision 51.5%, recall of 22.3% and F_1-score of 31.1%. The analysis shows that expansion of abbreviations does not improve outcome. By making expansion of some abbreviations and then stemming, the part that only has data in part 6, gives the best results of the study with precision 65.7%, recall of 59.9% and F_1-score of 62.7%. The results confirm that the classification of the care of central venous catheter gets better by looking at areas of nursing documentation as opposed to all content within the template. Results are also improved by performing preprocessing steps as expanding some abbreviations and stemming. We also found that the incident care of CVCs occurs most in part 6 of nursing template.

Innhold

Tabelloversikt

Figuroversikt

Problembeskrivelse

Når en pasient blir lagt inn på et sykehus skriver sykepleiere rapporter om pasienten i form av tekstlige dokumenter. Hvert enkelt dokument er lagd som en mal, og inneholder et bestemt sett med emneoverskrifter. De samlede sykepleierdokumenter for en pasient danner da sykepleiejournalen. I sykepleiejournalen beskrives også hendelser knyttet til sentralt venekateter (SVK), som er et utstyr som gir tilgang til pasientens blodbane. Sykepleiejournalen blir lagret i en database ved sykehuset. Denne databasen inneholder alle opplysninger om pasientens sykehusopphold, og omtales som sykehusets elektroniske pasientjournal (EPJ). I databasen finnes det informasjon som er strukturert og ustrukturert. Den strukturerte informasjonen kan vi søke frem ved hjelp av rapporter og andre hjelpemidler. Tekstlige dokumenter en sykepleier har skrevet er et eksempel på ustrukturerte data. For å kunne hente ut informasjon fra sykepleiejournaler må vi bruke andre metoder fra datateknologien som kan klassifisere dokumentene etter innhold. Vi bruker da maskinlæring og klassifikatoralgoritmer til dette formål.

Myndighetene og sykehusene er opptatt av å ha statistikker over pasienter som har hatt SVK i løpet av sitt sykehusopphold da SVK øker risikoen for infeksjoner. Opplysninger om pasienter som har hatt SVK finnes både i den strukturerte og ustrukturerte delen av EPJ. Det meste av informasjon om SVK finnes i den ustrukturerte delen av EPJ. Det er ikke lett å identifisere pasienter som har hatt SVK i løpet av sykehusoppholdet ved å lese gjennom journalen.

Evicare er et forskningsprosjekt hvor NTNU, Kunnskapssenteret, Dips ASA, Helse Innlandet og Akershus universitetssykehus(A-hus) deltar. Evicare har hentet ut tekstlige dokumenter for 800 sykehusopphold ved A-hus, og skal blant annet se på hendelser som er knyttet til SVK. Christine Tvedt ved Kunnskapssenteret har annotert deler av dokumentene med blant annet hensyn på stell av SVK.

Denne masteroppgaven skal lage en klassifikator for fritekst som kan avgjøre om et sykepleiedokument omtaler stell av SVK eller ikke. Klassifikatoren vil bli utviklet ved å bruke naturlig språkporsessering og maskinlæringsalgoritmer. Masteroppgaven studerer også sykepleiejournaler som inneholder annoteringer for stell av SVK, og vil se om det er visse emneoverskrifter som beskriver stell av SVK oftere enn andre deler av dokumentet. Ved å bruke kunnskapen om sykepleiedokumenter, vil vi se om klassifikatoren blir bedre ved å bruke innholdet i et emne fra sykepleiedokumentet i motsetning til innholdet innenfor alle emner. I tillegg vil vi se om preprosessering som ekspandering av forkortelser og stemming forbedrer klassifikatoren ytterligere.

1. Introduksjon

I det medisinske domenet brukes ulike datasystemer i forbindelse med behandling av pasienter. Det spenner seg fra spesifikke fagsystemer til elektronisk pasientjournal (EPJ), hvor all dokumentasjon av behandling for hver enkelt pasient skal lagres. De fleste systemene i helsevesenet lagrer en del av sine data strukturert i en database, mens resten blir lagret som tekstdokumenter, gjerne i den samme databasen. Dokumentene inneholder data om pasienten, behandlingen som er gitt og resultatet av behandlingen. I tillegg er dokumentene skrevet av ulike yrkesgrupper og strukturen på dokumentene kan variere ettersom hvilken yrkesgruppe som har skrevet dokumentet. Noen dokumenter vil være delvis strukturert, mens andre er ustrukturerte. De delvis strukturerte dokumenter kan være rapporter som er skrevet etter en mal. Dette gjør at informasjonen i ulike dokumenter varierer og informasjon vi er ute etter kan finnes i ulike dokumenttyper. En lege vil kunne bestemme at en pasient skal få antibiotikabehandling i 10 dager, mens det er sykepleierne som gir pasienten medisinen. En lege vil da dokumentere at pasienten skal få antibiotika og vil dokumentere dette i et journalnotat som legene oppretter. En sykepleier vil da gi pasienten medisinen og beskrive hvordan medisinen er gitt, når den er gitt og dokumenterer dette i et sykepleiedokument.

Ved sykehus er bruken av sentralt venekateter omfattende, og det er knyttet stor risiko for en pasient å ha dette utstyret (O'Grady, Alexander et al. 2002). Det er stor interesse for både sykehus og myndighetene, å ha oversikt over bruken av SVK(Pasientsikkerhetsprogrammet 2014). Det viser seg at det å finne informasjon om pasienter som har SVK er utfordrende ved å lete gjennom den strukturerte delen av en elektronisk pasientjournal (Penz, Wilcox et al. 2007)(Vedlegg 1). Forskningsprosjektet Evicare skal blant annet identifisere pasienter som har SVK ved bruk av ustrukturert tekst. Evicare har annotert deler av materialet som er hentet ut, og sett spesielt etter om stell av SVK er nevnt i dokumentene. Christine Tvedt ved Kunnskapssenteret har gjort denne jobben i Evicare. En tilnærming til å identifisere pasienter med SVK er å se på sykepleierdokumenter, og deres dokumentasjon på stell av SVK. Denne masteroppgaven skal finne ut om det er en del av sykepleiemalen som omtaler stell av SVK mer enn andre deler. Hvis så er tilfelle, skal vi ved bruk av dokumentklassifisering se om klassifikasjonen blir bedre ved å bruke data fra denne delen mot alt innhold innenfor hele malen.

Gjennom min jobb ved Akershus universitetssykehus (A-hus) henter jeg ofte data fra fagsystemer på sykehuset som grunnlag til forskning på sykehuset. I de fleste tilfeller hentes det ut strukturerte data fra de ulike fagsystem, som sammenstilles og gis til forskerne. I det siste har det blitt mer etterspørsel etter informasjon fra ustrukturert tekst, noe vi har vanskeligheter med å tilby. Som mastergradsstudent ved NTNU innenfor helseinformatikk og prosjektdeltaker i forskningsprosjektet Evicare er jeg interessert i å lære mer om muligheter til og bedre utnytte informasjon som finnes i tekstlige dokumenter. Jeg er spesielt interessert i å lære om prosessering av ustrukturert tekst og ulike teknikker for å gjøre dette, samt om dokumentklassifisering, og deres ulike læringsalgoritmer.

1.1. Bakgrunn

Evicare er et forskningsprosjekt hvor NTNU, Kunnskapssenteret, Helsebiblioteket, Dips ASA, Sykehuset Innlandet og Oslo Universitetssykehus deltar. Prosjektet er delt inn i ulike arbeidspakker (WP). Denne masteroppgaven inngår som en del av forskningsprosjektet Evicare, arbeidspakke 6 (WP6). WP6 er et samarbeid mellom Kunnskapssenteret, NTNU og A-hus.

Arbeidspakke 6 skal ut fra tekstlige journaler blant annet finne hendelser om sentralt venekateter (SVK). Bakgrunnen for dette er at det er vanskelig å finne informasjon om pasienter som har hatt SVK på ett eller annet tidspunkt ved et sykehusopphold, og at det er knyttet stor risiko for infeksjoner ved å ha SVK i kroppen.

Evicare skulle i utgangpunktet ha data levert i tre faser. Fase 1 skulle ha data for 50 pasienter med SVK og 50 pasienter uten SVK for annotering. A-hus bruker det elektronisk pasient-journalsystemet (EPJ) Dips. Når en pasient blir innlagt ved A-hus, kan data lagres som fritekst i dokumenter eller som søkbare felter i databasen. Eksempler på søkbare felter er diagnoser, prosedyrekoder og avdeling pasienten tilhører. For å oppfylle kravene i fase 1 forsøkte vi å identifisere pasienter med SVK ved å bruke prosedyrekoden, PYGC00. Prosedyrekoden PYGC00 betyr «Innlegging av sentralt venekateter». Vedlegg 1 viser søk som er gjort etter prosedyrekoden for SVK og treff denne fikk i søk etter forekomster av denne koden. Det viste seg at det kun er 198 pasienter som har operert inn SVK ved A-hus siden Dips ble innført som EPJ ved A-hus. Dette er et alt for lavt tall. Vår oppdagelse her stemmer ganske godt med følgende påstand fra Penz, Wilcox et al. (2007): «man finner flere personer som har SVK ved å lete i tekst enn i de administrative systemer». Dette førte til at vi ikke kan basere uttrekk på prosedyrekoder, da man ikke med sikkerhet kan gruppere data etter pasienter med og uten SVK.

Det å hente ut dokumenter fra Dips har vist seg å være utfordrende på flere måter. Et journaldokument skrives i en editor som bruker rikt teksformat (RTF). Når dokumentet skal lagres i databasen, lagres dokumentet med formateringen fra RTF i en spesiell type dataelement (blob/clob). Når vi skal hente ut data fra disse elementene er det begrensinger på hvor mye data man kan hente ut, samt at formateringen følger med. Formateringen er da også forandret slik at en editor ikke kjenner igjen formatet. En følgefeil er at noen typer dokumenter, som epikriser, ble uten innhold etter uthenting. Vi fikk hjelp av Dips til å forsøke å fjerne formateringer, men dette ga ikke resultater. Etter å ha lett på nett fant vi en måte å hente ut dokumenter på fra disse dataelementer slik at formateringen ble fjernet. Dette krevde spesielle løsninger som ikke er lett å få implementert i en organisasjon som A-hus. For oss ser det ikke ut til at noen har hentet ut innhold i dokumenter fra Dips tidligere. Et annet problem vi har oppdaget er at en del dokumenter er veldig like, hvor det kun skiller noen ord eller en setning mellom dokumenter. Det har vist seg at Dips skiller på dokumenter om de er aktuelle eller ikke. Det er en måte å holde orden på ulike versjoner av dokumentet. Så langt vi har forstått dette er dokumentet aktuelt dersom det er godkjent/signert av helsepersonell. Dette referer til et felt i databasen som heter recordtype. Typisk verdi for dette feltet er 0, som er

aktuelt og 1 som ikke er aktuelt dokument. Da disse opplysningene kom etter at annoteringen startet har datagrunnlaget blitt endret i løpet av studien. Hvordan dette er løst vises senere i denne delen av oppgaven.

For å kunne levere data til Evicare måtte vi ta i bruk andre dataregistre ved A-hus. Alle sykehus i Norge er pålagt å foreta to prevanelsregisteringer av infeksjoner og antibiotikabruk hvert år. På angitt dato og klokkeslett må sykehuset for alle inneliggende pasienter, som har infeksjon eller som får antibiotika, registrere visse opplysninger som de skal sende inn til Norsk Folkehelseinstitutt. Et av spørsmålene det skal svares på i registreringen er om pasienten har SVK eller ikke. A-hus foretar fire ganger i året denne typen registrering. Vi tok utgangspunkt i en av disse registeringer og brukte denne for å hente ut data til Evicare. Vi identifiserte alle pasienter som var innlagt på prevalensdatoen gjennom Dips. Vi identifiserte deretter hver enkelt pasients sykehusopphold som var innenfor prevalensdato, og hentet ut alle tekstlige dokumenter knyttet til aktuelt sykehusopphold. Resultatet av dette var at vi da hadde et datasett med 677 pasienter(innleggelser) hvor minst 29 pasienter hadde SVK på prevalensdatoen. Vi mente at antall pasienter med kjent SVK var for lite og bestemte oss for å øke antallet pasienter med kjent forekomst av SVK. Disse ble identifisert ved hjelp av tidligere prevalensregistreringer ved A-hus. På samme måte som tidligere hentet vi ut alle dokumenter for pasientens sykehusopphold hvor pasienten var registrert med SVK. En tabell med oversikt over prevalensregistreringene det er hentet ut data fra vises i tabell 1. Jeg omtaler personvern i forbindelse med uthenting av data i del 3.6 i denne oppgaven.

Vi kan nå skille pasienter som har hatt SVK fra de som ikke har hatt SVK. Det er viktig å merke seg at vi ikke kan si at antall pasienter med SVK er fastsatt da prevalensregistreringen sier noen om status for et gitt tidspunkt. Hva som har skjedd med pasienter etter prevalensregistreringen vet vi ikke noe om når det kommer til SVK.

Tabell 1: Oversikt over uttrekk av notater til annotering

Prevelansnummer	Antall filer hentet ut	Aktuelt dokument (recordtype = 0)	Ikke aktuelt (RecorType = 1)
1	2147	1980	167
2	32104	30332	1772
3	2808	2680	128
4	2708	2521	187
5	2883	2678	204
6	1369	1265	104
7	1595	1470	125
Sum	45614	42926	2687

Evicare har hentet ut 45612 tekstlige dokumenter til bruk i studien. Når vi tar hensyn til recordtype ser vi at det totale antall tekstlige dokumenter i prosjektet er på 42926. Dokumentene som er hentet ut fra Dips ble navngitt på følgende måte:

Løpenr_ID-nr_InnleggelsesID_Dokumenttype_Tidsstempel.txt

Christine Tvedt ved Kunnskapssenteret har annotert deler av tekstdokumentene med hensyn på SVK hendelser angitt i tabell 2.

Tabell 2: Merker på forekomster av SVK

1. Velg annotering CareCVC
Carecvc
CVC
Device
Hickman
Inscvc
InsertionSite
JugularVein
Other
PlanCarecvc
PlanInscvc
PlanRemcvc
PossibleCVC
Remcvc
Sepsis
SubclavicanVein
Symptom
Veneport

Fra tabell 2 ser vi at det er ulike merker som knyttes til teksten i ulike dokumenter. En kort beskrivelse av merkene er som følger:

- Carecvc: Angir om teksten kan klassifiseres som stell av SVK
- CVC: Angir om SVK finnes
- Device: Angir hvilken type SVK som er brukt
- Hickman: En type SVK
- Inscvc: Angir om SVK blir/er satt inn
- InsertionSite: Angir hvor SVK er satt inn.
- JugularVein: SVK er satt inn i jugular vein
- Other: Merke for andre opplysninger
- PlanCarecvc: Angir at det er planlagt stell av SVK senere
- PlanInscvc: Angir at det er planlagt å sette inn SVK
- PlanRemcvc: Angir at det er planlagt å fjerne SVK
- PossibleCVC: Angir at det er mulig pasienten har SVK
- Remcvc: SVK har blitt/er fjernet
- Sepsis: Pasienten har sepsis.
- SubclavicanVein: Angir at SVK er plassert i Subclavian vein.
- Symptom: Angir at det er symptom for blodbaneinfeksjon (BBI)
- Veneport: En type SVK

Tabell 3: Annoteringer.

Prevelansnummer	Antall filer annotert (Hvor tom ann fil angir ingen funn)	Annoterte filer med innhold	Andel annoterte filer hvor RecordType = 0 (Korpus)	Andel annoterte filer hvor RecordType = 1 (ekskludert)	Andel av annfiler med innhold
1	2147	289	1980	167	266
2	8668	951	8204	464	904
3	2804	341	2676	128	334
4	2708	377	2521	187	355
5	2883	432	2679	204	394
6	1369	165	1265	104	153
7	1595	190	1470	125	174
Sum	22174	2745	20795	1379	2580

Av tabell 3 ser vi at det er annotert 22174 filer, men at det etter korrigering for recordtype er totalt 20795 filer i korpus til Evicare. Av disse filene er det annotert funn av SVK hendelser i 2580 dokumenter, resten av dokumentene er negative. Annoteringen i Evicare er gjort ved å bruke programmet Brat. Christine Tvedt har annotert materialet, Laura Slaughter og Hans Moen har hjulpet til med oppsettet av Brat. Ved funn av SVK hendelse i et dokument, markeres teksten og det angis tag til teksten. Et eksempel på hvordan innholdet er kan være som følger:

«T1 CareCVC 635 649 Kommer med CVK»

T1 er en teller for antall merkinger som er gjort for dette dokumentet. Selve tag kommer som neste punkt med en posisjonsangivelse på hvor i dokumentet teksten forekommer, og til slutt kopi av selve teksten som forårsaker annoteringen. Det er også verdt å merke seg at dokumentet skiller informasjonen fra hverandre ved å bruke tabulator som skilletegn mellom informasjonsdeler. Tekstfilene og annoteringsfilene heter det samme slik at vi kan skille annoteringer fra tekstfilene.

På bakgrunn av data innsamlet i Evicare skal denne oppgaven lage en klassifikator for stell av SVK (CareCVC) i sykepleiedokumentasjon. På bakgrunn av annoteringene er det en oppfattelse om at stell av SVK forekommer mest i sykepleiedokumenter. Denne oppgaven skal ved en manuell gjennomgang av annoteringer for CareCVC finne ut andelen av annoteringene som er innen sykepleiedokumenter. Vi må først skille ut annoteringer om CareCVC, og deretter se hvilke av disse dokumenter som inneholder en eller flere emneoverskrifter fra sykepleiedokumenter. Videre må vi for sykepleiedokumentene identifisere hvilken del av malen annoteringene forekommer. Vi vil se om kunnskapen om hvor stell av SVK forekommer mest kan gi bedre resultat ved klassifisering. Vi vil videre se om preprosessering som ekspandering av forkortelser og stemming påvirker resultatet.

1.2 Forskningsspørsmål

Denne oppgaven består av to deler. En del vil på bakgrunn av annoteringer som er gjort i Evicare skaffe ny kunnskap om i hvilket emne, fra sykepleiemalen, stell av SVK forekommer. Del to vil ta utgangspunkt i emnet fra sykepleiemalen som omtaler stell av SVK mest. Vil tekst innenfor dette emnet gjøre at klassifisering av stell av SVK blir bedre enn ved å bruke all tekst innenfor sykepleiemalen. I tillegg vil vi se om preprosessering som ekspandering av forkortelser og stemming vil forbedre resultatene ytterligere.

Denne masteroppgaven forsøker å finne svar på følgende spørsmål:

- Vil klassifisering av stell av SVK bli bedre ved å utnytte kunnskap om hvor i sykepleiemalen stell av SVK forekommer?
- Kan preprosessering som ekspandering av forkortelser og stemming forbedre klassifiseringen ytterligere?

1.2. Begrensninger

Dokumentklassifisering berører områder som informasjonsekstraksjon, maskinlæring, informasjonsgjenfinning, Naturlig språkprosessering og data mining. Hvert av disse områdene er omfattende og krysser inn i hverandre. Den tiden som er tilgjengelig for denne oppgaven gjør at noen prioriteringer og begrensninger må velges.

- Da vi skal se på om klassifiseringen av dokumenter blir bedre av å bruke kun en del av dokumentet mot hele innholdet innenfor malen, vil vi kun se på bruk av en algoritme for klassifisering. Algoritmen Naive Bayes er valgt til dette formålet.
- Da vi skal bruke en algoritme er også datasettene fordelt i to grupper (test og trening), istedenfor tre; trening, utvikling og test. Grunnen til dette er at vi skal trene med en algoritme, og ikke flere. Hvis vi skulle trent med flere algoritmer ville vi da valgt å trene med treningsdata, teste alle algoritmer på utviklingsdata, for så å velge den som gir best resultat på utviklingsdata, for så å foreta endelig test av denne algoritmen på test data.
- Det er gjort preprosessering av data som forkortelser og stemming. Det er på ingen måte en komplett gjennomgang av forkortelser og stemming, men et forsøk på å se om dette gir resultater.
- Vi har valgt å ikke foreta stavekontroll av korpus da det finnes mange språk i sykepleierdokumentasjonen. I tillegg er det utstrakt bruk av medisinske ord og uttrykk som ikke finnes i en vanlig ordliste.

1.3 Oversikt over oppgaven

Denne delen gir en kort oversikt og beskrivelse av de ulike kapitler i oppgaven består av:

Kapittel 2, Begrepsavklaringer og definisjoner: Oversikt over viktige begreper og definisjoner brukt i denne oppgaven.

Kapittel 3, Helseinformatikk: Dette beskriver kort ulike aspekter ved helseinformatikk og ulike områder innenfor helsedomenet som vi anser som viktige for oppgaven.

Kapittel 4, Dokumentklassifisering: Vi ser her på viktig teori innenfor dokument klassifisering, og ulike teknikker vi kan bruke i denne oppgaven.

Kapittel 5, Tilgjengelige ressurser og verktøy: Detter er programmer som hjelper til med å foreta dokument klassifisering.

Kapittel 6, Relatert arbeid: Denne delen viser tidligere arbeider som er gjort innenfor dokument klassifisering av sykepleiedokumenter.

Kapittel 7, Design av eksperiment: Vi ser her på design av hvordan oppgaven skal løses.

Kapittel 8, Eksperiment: Vi ser her på hvordan eksperimentet er løst.

Kapittel 9, Resultater: Vi ser her på resultater av eksperimentet.

Kapittel 10, Evaluering og diskusjon: Vi ser her på resultatene vi har oppnådd, og hva som kunne vært gjort annerledes

Kapittel 11, Konklusjon: Konklusjon av denne oppgaven.

Kapittel 12, Videre arbeider: Noen forslag til videre arbeider som har kommet frem gjennom denne oppgaven.

2. Begrepsavklaringer og definisjoner

SVK	Sentralt venekateter. Utstyr for å sikre vaskulær tilgang i ulike medisinske behandlingssituasjoner. Engelsk forkortelse er SVK.
PVK	Perifert venekateter.
SVK-BBI	Blodbaneinfeksjon som resultat av innlagt SVK.
EPJ	Elektronisk pasientjournal.
BBI	Blodbaneinfeksjon
Egenskap	Egenskaper(feature) som er viktige når det kommer til klassifisering.
Egenskapsuthenter	Programmet som henter ut egenskaper vi er interessert i å studere(Feature extractor).
NLP	Naturlig språkprosessering (Natural language processing).
Setningssplitting	Prosessen med å kunne finne setninger i en tekst.
Pipe-line	Et sett med prosesser som følger etter hverandre for og for eksempel kunne hente ut informasjon fra et dokument.
Token/tokenisering	Token er å splitte ord, tall, datoer fra hverandre. I norsk er skillet mellom ord mellomrom.
Stemming	Stemming er å fjerne endelser fra ord, slik at vi kun står igjen med stammen til ordet. Det brukes enkle ikke-grammatikalske regler og stammen behøver ikke være et lovlig ord.
Lemmatisering	Lemmatisering er å bruke bøyningsregler for å danne en grammatikalsk riktig grunnform av ordet..
Word sense disambiguation	Finne et ords betydning i en setning.
Part of speech tagging(POS-tagging)	Dette er en funksjon som forsøker å knytte verb, substantiv og lignende til ord som finnes i en setning.
Korpus	En samling dokumenter.
Bibliotek	Alle ord som forekommer i korpus
Term frequency	En liste over alle ord som forekommer i korpus, og deres respektive frekvens i korpus.
TF-IDF	Term frequency – inverse document frequency. Et mål på hvor godt en term er egnet til å skille enkeltdokumenter fra en mengde dokumenter.
Accuracy	Andel av korrekt predikerte verdier.
Precision	Andel av korrekt predikerte verdier som er sanne
Recall	Andel av korrekt predikerte verdier som systemet har definert som sanne.
F_1-score	Vektet gjennomsnitt av precision og recall.
Egenskapsområde (Feature space)	Tekst innenfor et eget område i et dokument.
Normalstatus	En dokumentmal som inneholder 15 emner som en sykepleier skal dokumentere etter.
Feature engineering	Finne nye egenskaper som vil hjelpe algoritmen til å klassifisere bedre.
Prevalens	Opptelling av hvor mange individer i en bestemt gruppe som har en gitt tilstand på et angitt tidspunkt
PYGC00	Prosedyrekode for innleggelse av sentralt venekateter

3. Helseinformatikk

Denne oppgaven kommer som et resultat av et videreutdanningsløp innen helseinformatikk ved NTNU. Det er naturlig å berøre helseinformatikkområder som er sentrale for denne oppgaven.

3.1 Pasientjournal

Når vi skal se på dokumenter i helsevesenet, spesielt på sykepleiejournaler er det viktig å forstå hva en pasientjournal er, hvordan den er bygd opp og hvordan dette er gjort i en elektronisk versjon av journalen. I følge Enrico Coiera (Coiera 2003) er hovedformålet med en pasientjournal at den effektivt skal kommunisere mellom ulike profesjoner i helsevesenet. I tillegg har journalen en standard struktur. Det er videre fire vanlige oppbygninger.

1. Integrert journal
   Her er data presentert i en kronologisk rekkefølge. Hver episode er angitt med dato og tid.
2. Kilde-orientert journal
   Her er journalen organisert etter hvem som genererte data. Det er i tillegg egne deler av journalen som er dedikert til medisinske notater, sykepleierdokumentasjon, labratoriesvar og så videre.
3. Protokoll orientert pasient journal
   Dette er systemer som har predefinert behandlingsløp for kjente sykdommer. Data som skal lagres er forhåndsdefinert og viser hva en lege skal hente inn av informasjon og behandlingsplanen for dette forløpet.
4. Problem orientert journal (POMR)
   Denne type journal har fire deler: en problemliste, initial plan, en database med pasientens data og en egen del for resultat av behandlingen. POMR organiserer data etter pasientens liste av problemer. Behandlingsplanen beskriver hva som skal bli gjort for hvert problem.

3.2 Den papirbaserte journal

Den papirbaserte journalen har eksistert i lang tid og har vært til god hjelp i den medisinske hverdag (Coiera, 2003 kapittel 10) (Coiera 2003). Selv om den fysiske journalen har vært den samme i lange tider, har strukturen på journalen endret seg mye de siste 50 år. Journalen har gått fra å være ustrukturert og kronologisk til å bli problemorientert eller oppgaveorientert. Den fysiske journalen har både fordeler og ulemper.

Fordelene er at den er portabel, alt som trengs er penn og papir for å legge til informasjon. Tilgangen til en fysisk journal føles bedre når det kommer til å se gjennom informasjonen som finnes.

Ulempene er at journalen kun kan brukes av en person av gangen, journalen tar opp fysisk plass, og papir er et usikkert medie. Papir kan fort gå i stykker eller man kan miste sider av journalen ved transport. I tillegg kan gjenfinning av informasjon fra en papirjournal være vanskelig. Det vises til en undersøkelse av 168 polikliniske konsultasjoner som ble gjennomgått for å lete etter bestemt informasjon, men informasjonen ble ikke funnet i 81 % av tilfellene. Informasjonen de trengte var alt fra laboratoriesvar til medikasjon. Det er også vanskelig å ha oversikt over hvem som har sett på en journal ved bruk av papir.

3.3 Elektronisk pasientjournal (EPJ)

En av de store fordelene et elektronisk journalsystem har er at man kan lagre store mengder data på et lite fysisk område. Data i EPJ er lett å dele, og tilgangen kan enkelt reguleres ved brukertilganger. Gjennom brukertilganger og logg systemer, er det også mulig å føre oversikt over hvem som har sett på informasjon om en pasient. Hvis vi tar dette videre vil vi kunne la datamaskinen hjelpe til med å gi alarmer og påminnelser ved behandling.

3.4 Sykepleierdokumentasjon

Da denne oppgaven bruker sykepleiedokumenter for å undersøke om disse inneholder informasjon om stell av CVK er det viktig å forstå litt av hva denne dokumentasjonen innebærer.

Kravet til at sykepleiere skal dokumentere planlagt og utført stell er hjemlet i ulike lover. Det er også krav til at dokumentasjonen skal gjøres elektronisk. I 1999 kom lov om helsepersonell og pasientrettighetsloven. Lov om helsepersonell krever at de som utøver helsehjelp skal dokumentere den hjelp de gir til pasienten. Pasientrettighetsloven gir pasienter rett til en individuell plan og medvirkning på utforming av planen. Forskrift om pasientjournal kom i 2001, og inneholder krav til hva en sykepleier skal dokumentere. En pasients normalstatus skal dokumentere pasientens tidligere helseproblem, funksjonsnivå og pasientens egen opplevelse av helsetilstand før innleggelse i sykehus. Normalstatus er en 14 punks liste over emner som det kan registreres data i (Sørli 2007).

Normalstatus brukes som dokumentasjon i stor grad ved A-hus for sykepleiere. Normalstatusen som brukes ved A-hus er gjengitt i tabell 4 under.

Tabell 4: Inndeling av normalstatus ved A-hus.

1 Kommunikasjon/sanser
2 Kunnskap/utvikling/psykisk
3 Åndedrett/Sirkulasjon
A Åndedrett
B Sirkulasjon
4 Ernæring/væske/elektrolyttbalanse
5 Eliminasjon
6 Hud/vev/sår
7 Aktivitet/funksjons-status
8 Smerte/søvn/hvile/velvære
9 Seksualitet/reproduksjon
10 Sosialt/planlegging av utskrivelse
11 Åndelig/kulturelt/livsstil
12 Annet/legedelegerte oppgaver

3.5 Sentralt venekateter (CVK/SVK)

Intravaskulære kateter, som sentralt venekateter (CVK) og perifere venekateter (PVK), er uunnværlig i et moderne sykehus. Sentralt venekateter er et medisinsk kateter som brukes for å gi tilgang til blodbanen i ulike behandlingssituasjoner. Som pasient er det en økt risiko ved å ha en CVK i kroppen i forhold til spredning av mikroorganismer i blodbanen og kateterrelatert blodbaneinfeksjon (CVK-BBI) (O'Grady, Alexander et al. 2002). PVK er utstyr som blir mest brukt for å gi vaskulær tilgang til blodbanen. Selv om PVK ikke har stor insidens av CVK-BBI er det stor dødelighet relatert til PVK som følge av den utstrakte bruken. I tillegg er det store kostnader målt i penger knyttet til pasienter som har fått blodbaneinfeksjon som følge av CVK.

Det finnes egne prosedyrekoder som leger bruker når de gir en pasient CVK. Ved å bruke prosedyrekoden for å lete etter pasienter med CVK vil vi ikke finne mer enn rundt 11 % av pasientene (Penz, Wilcox et al. 2007). Vi søker da etter informasjon som er lagret i en strukturert databaser. Det lagres heller ikke noen andre opplysninger om CVK i den strukturerte delen av EPJ. Av dette følger at vi ikke kan si noe om hvor lenge pasienten har hatt CVK eller om hvor ofte disse er stelt og sett til ved å bruke strukturerte kilder. Vi må da se nærmere på den ustrukturerte delen av EPJ for å finne svar på disse spørsmål.

3.6 Personvern

Personvern er et område som er viktig for Evicare og meg personlig som jobber ved A-hus. I min jobb ved A-hus har jeg delegert databehandlermyndighet, noe som setter klare grenser på hva jeg kan gjøre i ulike prosjekter. Evicare er et forskningsprosjekt hvor jeg også er deltaker. Jeg er i tillegg leverandør av data til Evicare prosjektet, i rollen som databehandler. I tillegg skal jeg være student, som er nok en rolle. Dette er grunnen til at jeg vil si noen ord om personvern og rolleavklaringer. Evicare, arbeidspakke 6, har godkjenning fra Regional Etisk komite og personvernombud ved A-hus. Mastergradsprosjektet er sendt inn som en endringsmelding til REK. I tillegg er det utarbeidet databehandleravtaler mellom A-hus og Kunnskapssenteret, og mellom A-hus og NTNU. Vedlegg 2 viser min taushetserklæring som student ved NTNU.

4. Dokument klassifisering

I dette kapitel utdypes metoder og hjelpemidler for å kunne klassifisere dokumenter som inneholder ustrukturert eller delvis strukturert tekst. Jeg vil bruke kunnskapen fra dette kapittel for oppbygging av mitt eksperiment med å kunne klassifisere sykepleierdokumenter som inneholder informasjon om stell av CVK.

En definisjon av dokument klassifisering er hentet fra Wikipedia (Wikipedia):

«Document classification or document categorization is a problem in library science, information science and computer science. The task is to assign a document to one or more classes or categories. This may be done manually or algorithmically. ”

Dokumentene som skal klassifiseres kan være av ulik type. Eksempler på dokumenttyper kan være tekst, avisartikler, behandlingsnotater, e-poster og så videre. Hver type dokument har sine spesielle egenskaper ved seg som gjør klassifiseringen utfordrende.

En mer formell måte å definere dokument klassifisering på er som følger (Jurafsky):

Input: Ett sett dokumenter d

Klasser: ett bestemt sett med klasser C = $\{c_{1,}\ c_{1},\ldots c_{j}\}$

Utdata: en beregnet klasse c Ɛ C

En måte å gjøre dette på er ved å bruke håndskrevne regler. Håndskrevne regler kan gi bra resultater hvis vi har en domeneekspert som annoterer materialet. En stor ulempe med hånsskrevene regler er at det koster mye tid å vedlikeholde reglene.

Dokumenter kan klassifiseres etter innhold eller etter andre egenskaper ved dokumentene, som forfatter eller årstall for publisering. Klassifiseringen kan godt gjøres av mennesker, men når antall dokumenter eksploderer er det en stor fordel å kunne gi jobben til en datamaskin. Når det gjelder klassifisering av dokumenter etter innhold er det to filosofier som er gjeldende: Innholdsbasert og Spørrebasert klassifisering.

Innholds basert klassifisering:

Klassifiseringen avgjøres basert på en form for vekting av innholdet i dokumentet. For automatisk klassifisering kan man telle forekomst av et ord. Hvis forekomsten er over en grense får den en klassifisering, hvis den er under en annen klassifisering.

Spørrebasert klassifisering

Dette er klassifisering som på bakgrunn av en forventet forespørsel fra en bruker avgjør hvordan dokumentet blir klassifisert.

Klassifisering er oppgaven med å sette korrekt merkelapp på inndata. Generelt er hver inndata behandlet for seg selv og merkene vi skal sette er definert på forhånd. Automatisk klassifisering deles gjerne inn i tre grupper:

1. *Supervised document classification:*
2. *Unsupervised document classification*
3. *Semi-supervised*

Ved automatisk klassifisering bruker vi som regel datamaskiner for å gjøre jobben, og oppgaven da er å finne kriterier for beslutning om hvilken klasse inndata tilhører. Denne måten omtales også som statistisk tekst klassifisering. Jeg kommer tilbake til mer beskrivelse av disse klassifiseringsalternativer i del 4.6. Klassifisering av dokumenter kan gjøres fra å bruke manuelle rutiner til å bruke en datamaskin. Ved å bruke en datamaskin finnes det ulike måter å gjøre dette på. Da dokumentklassifisering berører ulike områder som maskin læring, kunstig intelligens og andre områder, er det her valgt ut områder som anses som viktige for å kunne klassifisere sykepleierdokumenter med hensyn på stell av CVK.

4.1 Velge egenskaper for klassifikatoren (feature selection)

Ett av de første stegene man må gjøre for å lage en klassifikator er å bestemme hvilke features som er relevante for oppgaven, og hvordan egenskapene skal merkes. Et eksempel på en egenskap er siste bokstav i et navn. Funksjonen som henter ut siste bokstav i et navn vil da bli benevnt som feature extractor og biblioteket over alle siste bokstaver i navn blir omtalt som feature set. Feature sets verdier er som regel boolske, tall eller tekststrenger (Bird, Klein et al. 2009).

Det å velge riktige features og hvordan disse skal bli kodet spiller en stor rolle i forhold til hvor god modell læringsalgoritmen lager. Å velge features er en prosess som består av prøving og feiling. Det er vanlig å starte med å la alle egenskaper være med for så å finne de egenskaper som er til hjelp. Det er et problem hvis vi lar for mange egenskaper være med til å lage modellen. Modellen vil da ikke generalisere godt nok for nye usette data. Dette problemet blir omtalt som overfitting, og er spesielt problematisk når det kommer til små datasett.

4.2 Natural language processing (NLP)

De dokumenter som vi skal klassifisere foreligger i elektronisk format og er skrevet på norsk. Det er viktig å kjenne til hvilke muligheter som finnes for å behandle språk maskinelt. Denne delen belyser hvilke muligheter som finnes ved å bruke naturlig språkprosessering.

Det finnes mange ulike typer språk (norsk, japansk, C eller Cobol). Norsk er et eksempel på naturlig språk. Egenskapen til et naturlig språk er at det endrer seg over tid og at det er vanskelig å tilrettelegge eksplisitte regler (Bird, Klein et al. 2009). Motsetningen er kunstige språk, som et programmeringsspråk som sjelden endrer seg, og som er eksplisitt. Før vi kan trekke konklusjoner eller kunnskap ut av mange tekstlige dokumenter er det flere programmeringsmessige steg som bør foretas, dette er ofte referert til som en pipeline. Selv om en har fått til gode resultater i et domene, for eksempel noveller skrevet på 1800 tallet, er det ikke nødvendigvis slik at denne jobben vil fungere bra for web-dokumenter. Nadeau og Sekine peker på at jobber som er gjort innenfor et domene ikke automatisk kan flyttes over til et annet domene ellet tekstlig genre (Nadeau and Sekine 2007).

Mye av tilgjengelig kliniske data i dag er i form av tekstlige dokumenter som har kommet frem som følge av transkripsjon av diktater, skrevet av behandlere eller som et resultat av talegjenkjenning.

Det som skiller kliniske tekster med for eksempel biomedisinsk tekst er følgende:

- Setningene er ikke grammatisk riktig (korte setninger)
- Setningene inneholder mange forkortelser
- Det er mye skrivefeil
- Det finnes kopiert tekst fra for eksempel prøvesvar
- Ulike forsøk på å standardisere tekster(Meystre, Savova et al. 2008).

Dette er viktige punkter å forholde seg til når man skal forsøke å klassifisere elektroniske tekster.

For å redusere feil og forbedre kvalitet når det kommer til tekstlige filer er annoterte data å foretrekke. Annoteringen består i å merke de dokumenter hvor informasjon om for eksempel CVK finnes. Det å annotere tekster er kostbart både når det kommer til forbruk av tid og til bruk av domeneeksperter.

Naturlig språk prosessering har foregått siden starten av 1950 tallet (Wikipedia). Naturlig språkprosessering (NLP) kan være alt fra å telle frekvens av ord for å sammenligne ulike skrivemåter, til å forstå alle språk og gi en meningsfull tilbakemelding på en spørring. Nadeau og Sekine peker på at vi må ta hensyn til domenet vi skal jobbe i. Jeg skal jobbe i det medisinske domenet og dette må det tas hensyn til.

Vi ser i denne delen av dokumentet på noen av prosessene som kan brukes i det medisinske domenet.

4.2.1 Setningssplitting

Å kunne bryte opp en tekst i seksjoner og setninger er en viktig del av NLP.

Setninger skal en tro er ganske lett å bryte opp ved å se etter punktum, spørsmålstegn og utropstegn. Det er ikke så lett som det da punktum ikke bare brukes til å avslutte en setning, men kan også være en forkortelse. Det finnes ulike måter å løse dette på. I medisin er ofte dokumenter skrevet etter diktat. Dette fører til problemer som grammatikkfeil, skrivefeil og punktum satt på feil plass(Cho, Taira et al. 2003).

Cho et al. mener at man bør bryte opp medisinske tekster i segmenter/seksjoner, og at dette ikke tidligere har blitt gjort innenfor medisinske dokumenter. Det meste av tidligere klassifiseringer er gjort på helsefaglige tidsskrifter. I helsevesenet finnes det mange yrkersgrupper som dokumenterer sin jobb fortløpende. Disse dokumenter deles gjerne innen yrkesgruppe, men også mellom ulike yrkesgrupper. Det vil si at alle formularer innenfor medisinsk dokumenasjon er delt opp i tematiske deler. Det er i dag ikke en standard for hvordan dokumenter skal se ut innenfor det medisinske domenet. Hvis en ser på sykepleierdokumentasjon som vi har hentet ut i prosjektet, er disse delt inn i ulike deler, selv om malen de skal bruke ofte er brutt eller i noen tilfeller ikke eksisterer i det hele tatt. Cho et al. mener at ekstraksjon av informasjon vil bli bedre hvis en kjenner hvilke deler av en rapport som inneholder de opplysninger som vi er interessert i. En lege vil kanskje bare se på diagnoser som er i en epikrise. En automatisert avsnitts ekstraktor vil være en fordel innenfor det medisinske domene. Medisinske rapporter er sjelden strukturert i dag, noe som gjør at dokumentstrukturen er ukjent for datamaskinen. Cho et al utviklet en algoritme i to steg. Steg 1 er en regelbasert algoritme er å finne starten på en seksjon. Steg 2 ser etter seksjonens start og slutt. De oppnådde god score for å identifisere rett seksjon/segment, og hadde tilsvarende liten score for å ha misset seksjoner.

4.2.2 Tokenisering

Etter å ha hentet ut de tekstlige journaler for dette prosjektet, var første steg være å bryte opp all tekst i symboler(engelsk tokens). Symbolene kan være ord, tall eller punktum. Tokenisering gjør dette ved å finne grensene for ord(punktum, komma osv.). Utfordringer knyttet til ord deling er med språket teksten er skrevet på.

Språk deles inn i to kategorier, segmenterte og usegmenterte språk. Norsk vil falle inn under kategorien segmentert språk da vi hovedsakelig benytter mellomrom for å skille ord fra hverandre (Bird, Klein et al. 2009). Det et er viktig å være oppmerksom på at stedsnavn som New York i noen tilfeller vil bli to tokens, New og York. Vi vil gjerne at dette blir oppfattet som en token, New York. Dette kan løses ved at man senere slår disse to token tilbake til en. Vi har vanligvis en stor grad av samskriving av ord, eksempler på dette er orddeling, venekateter og setningssplitting. Kvaliteten av tokeniseringen er avgjørende for kvaliteten til komponenter som kommer senere i pipe-line (Bruce 2012).

Normalisering av tokens er en vanlig prosedyre i NLP. Normalisering betyr at man gjør om alle tokens til små bokstaver. Det vi oppnår da er at for eksempel pasient og Pasient blir en token. Vi gjør dette slik at datamaskinen skal oppfatte dette eksempelet som en token.

Vi kan også foreta stemming, det vil si å fjerne endelser fra ord, slik at vi kun står igjen med stammen til ordet. Nok et steg videre er lemmatization som er å gjøre om ord til sin opprinnelige form(Bird, Klein et al. 2009). Lemmatisering er et alternativ til stemming.

Lemmatisering følger gramatikkregler, noe stemming ikke gjør da klassifiseringen ikke tar hensyn til gramatikken.

Dette er ulike deler for normalisering, og som gjør at senere deloppgaver får en bedre kvalitet.

4.2.3 Stavekontroll

Ruch et al. (Ruch, Baud et al. 2003) viser til at det i journaldokumenter er opptil 10 % med feilskriving, noe som er mye i forhold til andre domener. Noen av disse feilskrivninger stammer fra egennavn som ikke er med i ordlister.

Stavefeil kan settes i to grupper. En gruppe er rett og slett skrivefeil, noe som kalles kontekst sensitiv stavekorreksjon. Den andre er ord som ikke finnes i ordboka, og kalles kontekst uavhengig stavekorreksjon. Dette vil helt sikkert være tilfelle i de dokumenter vi har hentet ut fra EPJ.

Det er bevist at det å bruke en ordbok med kontekst spesifikk ordliste vil øke treffsikkerheten (accuracy) med mellom 5,75-9,94 % (Coden, Pakhomov et al. 2005).

Det finnes også metoder for å finne uttaleavledet feilstaving som «string-edit-distance», Levenstein score og metaphone algoritmen.

4.2.4 Forkortelser

En stor del av dokumentasjonen helsepersonell foretar om en pasients sykehusopphold består av tekstlige dokumenter. I disse dokumentene finnes det mange forkortelser eller akronymer som ikke er forklart i dokumentet. Hua et al. (Xu, Stetson et al. 2007) har lagd en metode for å finne forkortelser.

Første steg i prosessen var å finne forkortelser i korpus, for deretter å lage en forklaring/beskrivelse av forkortelsene. Hua et al. brukte Unified Language System (UMLS) og MEDLINE abbreviation database (ADAM) som kunnskapskilder når du skal lage denne databasen. I tillegg til at forkortelser blir brukt, er forkortelsene ikke entydige. En forkortelse kan bety flere ting i det medisinske domene. Hua et al. viser et eksempel på at forkortelsen RA kan bety «Right atrium» og «rheumatoid arthritis». Dette skaper selvsagt en ekstra utfordring. Det kan se ut til at spesialiteten legen har spiller en stor rolle for å kunne forklare forkortelsene som blir brukt.

Hua anbefaler at en domeneekspert ser etter forkortelser og gir disse en riktig forklaring.

4.2.5 Word Sense Disambiguation (WSD)

I tekster er det vanlig at tokens(ord) forekommer flere ganger og at de har ulike betydning i hver sin setning. Det er da vanlig å lage et system som lagrer mening til ulike tokens. Token kan her være ord, forkortelser eller fraser. Slike systemer er referert til som WSD, og skal hjelpe til med å entydig gjøre ord i en kontekst. WSD er kritisk for et system som for eksempel skal hente ut informasjon fra fritekst.

Liu et al.(Liu, Lussier et al. 2001) lagde et system som automatisk lager et WSD system. Tidligere ble dette gjort manuelt, noe som krever store ressurser i form av mennesker. I tillegg er det vanskelig å vedlikeholde og kan ikke uten videre tas inn i et nytt domene. For å lage slike systemer baseres de på algoritmer fra Information Retrieval (IR). Liu nevner tre ulike algoritmer for dette formål: Naivere Bayes, Decision list method og ememplar-based method. Resurser de har brukt er UMLS metathesaurus, medline og data fra New York Presbyterian hospital.

En dypere forståelse av språket er å finne ut av «hvem gjorde hva til hvem». Det vil si å finne subjekt og objekt til verb. Et eksempel fra (Bird, Klein et al. 2009) er : «Tyvene stjal maleriene. De ble solgt».

For oss mennesker er det lett å se hva som ble solgt. For en maskin er dette ingen lett oppgave. For å kunne svare på dette må vi finne forløperen til «De» i andre setning, enten maleriene eller tyvene. Teknikker for å kunne svare på slike problemstillinger kalles «anaphora resolution» og merking av semantisk rolle.

4.2.6 Part of speech tagging (POS-tagging)

En POS tagger kan enkelt forklares som et program som kan bryte opp setninger i verb, substantiv, adjektiv og så videre. Figur 1 nedenfor viser, i de små boksene, hvordan en slik tagging kan se ut. POS-tagging algoritmer er vanligvis regelbasert eller stokastisk (Bird, 2009 , side 179). Dagens POS taggere er gode når det kommer til nøyaktighet(accuracy), men disse bruker da statistiske modeller som Hidden Markov Model (HMM) eller andre modeller. For disse modellene er det viktig at data er hentet fra samme domene og at størrelsen på data er over 1 million ord. Dette for at statistikken skal være til å stole på.

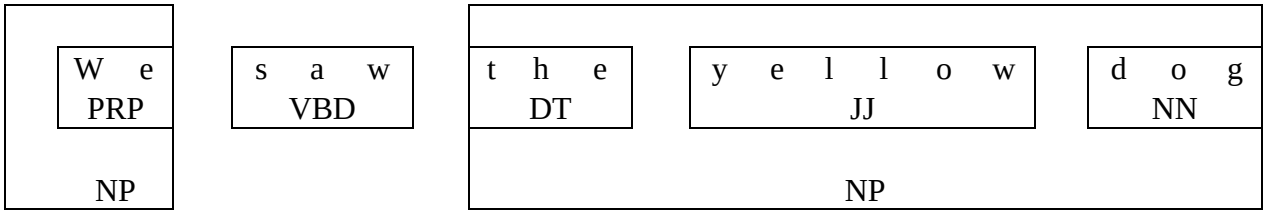
Det finnes mange POS-taggere i markedet i dag, både som åpen kildekode og hyllevare. Når en POS-tagger blir brukt på et nytt domene som i det medisinske, vil en forvente nye forekomster av ord (Coden, Pakhomov et al. 2005). Utfordringen her blir da at taggeren ikke vil bli nøyaktig nok. Coden et al. forsøker å kvantifisere forskjellene mellom domenet vanlig engelsk og domenet medisinsk engelsk, når det kommer til POS-tagging. De annoterer medisinske ord manuelt og legger dette til treningsdata for engelsk. Den andre metoden bygges det opp et leksikon, avledet fra det medisinske domenet. Coden et al. henter dokumenter fra det medisinske domene og tester med to metoder for POS tagging. Målet for Coden er å lage en POS tagger som gir bedre nøyaktighet(accuracy) på medisinske tekster. Entydighet(Unambigous) er definert som en enkelt tag er assosiert med et ord innen et sett med data, dette er også kalt hapaxes. Et ord er flertydig(ambiguous) hvis den har flere tags assosiert med seg. En høyere prosent med ambiguous ordtyper øker vanskeligheten med POS tagging. Ved å lage en ordliste på topp 500 ord i medisinsk domene og legge denne til engelsk domene øker accuracy med 5,75 – 9,94 %. Det er viktig å ta hensyn til dette. I følge Campell kan feil på 4 % i POS-tagging gi 10 % feil på setningsnivå.

Det er vanlig å dele inn i to leksikalske kategorier, åpen og lukket. Den åpne klassen gir mening til en setning (verb, substantiv, adjektiv og adverb). Den lukkede klassen inneholder et sett statiske ord, som ikke gir noen mening til en setning. Når en POS-tagger støter på et nytt ord, er det som oftest i den åpne klassen dette ordet hører til (Cho, Taira et al. 2003).

4.2.7 Chunking

For å kunne identifisere entiteter i en tekst brukes en teknikk som kalles chunking. Chunking brukes til å finne fraser for substantiv, verb og adjektiv.

Figur 1: Part of Speech tagging



Figuren ovenfor viser hvordan tokenisering, POS-tagging og chunking kan se ut (Bird, 2009, side 264). De små taggene som er under hvert ord er kommet frem som følge av POS-tagging, mens de store boksene som har benevnelsen NP kommer etter chunking. Chunking kan deles inn i flere underdeler. Den første er noun pharse chunking. Det er av og til lettere å ekskludere ord fra en chunk, dette er kalt chinking. Chunks har 2 måter å bli presentert på, som tagger (vist i figur 1) eller som trær. Det er verdt å merke seg at chunking er veldig avhengig av POS-tagging. Jo bedre POS-taggingen er, jo bedre blir resultatet av chunking.

Chunking kan gjøres på bakgrunn av håndskrevne regler eller som maskinlæring. Chunking bruker tags for å kunne samle et sett av ord som danner en frase. Taggene består av startfase, innenfor og sluttfase.

4.2.8 Named entity recognition and classification (NERC)

Named Entity (NE) er oppgaven med å finne bestemte substantiv fraser som refererer til spesifikke individer. Eksempel på slike individer kan være personer, organisasjoner, sted, dato og penger.

En av hovedoppgavene til Named entity recognition (NER) er å finne alle NE i en tekst. Dette kan gjøres på ulike måter. Det er vanlig å kunne kategorisere egennavn med at det starter med stor bokstav. Datoer og tid er viktig også egenskaper vi vil finne. Mange yrkestitler ender med ist (journalist, syklist), land slutter på sk (dansk, svensk). Å finne slike identiteter i en tekst er ansett til å være en av de viktigste deloppgaver i informasjonsekstraksjon.

David Nadeay og Satoshi Sekine (Nadeau and Sekine 2007) har gjennomgått siste års publikasjoner innen NER. De har sett på de tidligere års regelbaserte systemer til de senere års bruk av maskinlæringsteknikker. De fant ut at NERC fortsatt er konsentrert rundt et begrenset domene og genre som avisartikler og web sider. De fant også ut at NERC har gått fra å være håndskrevne regler til mer å gå over mot maskinlæring. I tillegg til å kunne dele inn ord fra et domene er det også viktig å kunne finne egennavn og numeriske entiteter som nevnt ovenfor.

4.2.9 Relasjonsekstraksjon

Entitetsrelasjoner gjøres etter NER. Oppgaven her er å lage relasjoner mellom entitetene. Det vi leter etter er setninger som har trippel setningsoppbygging. Det vil da være i form av navngitt entitet, samling tekster og enda en navngitt entitet (X, α , Y). Vi kan da bruke vanlige uttrykk for å hente ut bare de instanser av α som oppfyller våre krav(Bird, Klein et al. 2009).

4.2.10 Negasjon

Etter at entitetene er behandlet er det interessant å finne ut at opplysningene man potensielt kan trekke er korrekte. Vi må sjekke etter negasjoner og opplysninger som omhandler andre personer. Et eksempel på dette er «familien er ikke disponibel for kreft». Dette er en viktig parameter å sjekke da det er lett å ta feil beslutning.

4.3 Bag of words

Bag of words er en modell som blir brukt i NLP, information retrieval og dokument klassifisering. Modellen går ut på å samle begreper, i vårt tilfelle ord, i en egen liste. Denne listen vil da bestå av alle ord som forekommer i et dokument, eller en samling dokumenter. Listen er for øvrig uavhengig av rekkefølgen ordene er i dokumentet eller korpus. Ordene vil også representere en feature(egenskap) når det kommer til dokument klassifisering. I tillegg til å samle alle ord i korpus, telles også frekvensen av ord. Frekvensen kan da bli sett på som en vekt, og kombinasjonen mellom dokumenter og frekvens blir omtalt som term frequency. Term frequency skrives som $td_{f,d}$ (Manning, Raghavan et al. 2008, side 107). Listen over ord blir også omtalt som bibliotek (Wikipedia). Et enkelt eksempel på bag of words følger under.

Hvis vi har to dokumenter:

Dokument 1: Jeg liker ishockey

Dokument 2: Jeg liker ikke Ishockey

Vi vil da ende opp med et bibliotek som ser slik ut:

jeg: 2

liker: 2

ishockey: 2

ikke: 1

Dokumentene kan da representeres som en vektor, som ser slik ut for dokumentene:

Dokument 1: [1,1,0,1]

Dokument 2: [1,1,1,1]

4.4 TD-IDF

Term frequency har en utfordring når det kommer til at noen ord forekommer mange ganger i korpa, eller at korpa er veldig stor. For å håndtere dette problemet er tanken at vi heller reduserer vekten av term frekvensen med en faktor som vokser, men ikke så fort som frekvensen. Det er da mer vanlig å bruke dokument frekvensen av en term, anotert som df_t. Tanken er at det er bedre å skille dokumenter fra hverandre basert på statistikk på dokumentnivå, istedenfor hele korpa. Vi skalerer ned vekten av termen med funksjonen: $idf_t = log^N_{df(t)}$, hvor N er totalt antall dokumenter i korpus.

Vi kan nå kombinere definisjonen av term frekvens og invers dokument frekvens til å lage en vekt for hver term i hvert enkelt dokument med formelen(Manning, Raghavan et al. 2008, side 109):

$$tf.idf_{t,d} = tf_{f,d} \text{ x } idf_t$$

Vekten har noen egenskaper som det er verdt å merke seg:

- Vekten har egenskapen at termen er høyest hvis frekvensen er høy i et lite antall dokumenter.
- Vekten er lavere når termen forekommer færre ganger i et dokument, eller forekommer i mange dokument.
- Vekten er lavest når termen forekommer i nesten alle dokument.

Vi kan nå se på hvert dokument som en vektor. Elementene i vektoren tilsvarer termene i biblioteket. Verdien til elementene i vektoren er beregnet ut fra vekten til termen og frekvensen til termen i aktuelt dokument. Hvis termen ikke eksisterer i dokumentet vil elementet i vektoren være 0. Tilsvarende vil termer som forekommer x antall ganger i dokumentet bli tallet for elementet x ganger vekten til termen i aktuellt dokument. Vi kan beskrive dette på følgende måte:

$$\text{Score(q,d)} = \sum_{t \epsilon q} tf - idf_{t,d}$$

4.5 Vector space model.

Det å kunne representere et dokumentsett i ett felles vektor rom blir omtalt som vektor rom modellen. Vektor rom modellen er grunnleggende for områder som information retrieval og dokument klassifisering.

Vi beskriver vektoren for et dokument som $\vec{V}$(d) med et element i vektoren for hver term i biblioteket. Vektoren kan representeres i vektor rom hvor hver term representerer en akse. Det å sammenligne vektorer er utfordrende da to dokumenter ser like ut, men har ulik lengde(antall termer) og ulik frekvens av ord. Måten å kvantifisere likheten mellom to dokumenter på er å beregne cosine similarity: sim(d1,d2) = $\frac{\overrightarrow{V(d1)} * \overrightarrow{V(d2)}}{|\overrightarrow{V(d1)}||\overrightarrow{V(d2)}|}$

Her er da telleren dot product (inner product) av vektorene $\overrightarrow{V(d1)}$ og $\overrightarrow{V(d2)}$

Nevneren er produkt av deres Euclidian lengde, som beregnes på følgende måte: $\sqrt{\sum \overrightarrow{v^{2(d)}}}$

Dot product av to vektorer er definert som følgende: $\sum_{i=1}^{M} xy$

Vi kan da beskrive sim(d1,d2) = $\underset{v}{\rightarrow}(d1) * \overrightarrow{v(d2)}$(Manning, Raghavan et al. 2008, kapittel 6.3.1)

4.5.1 Vektor rom klassifisering

Vektor rom klassifisering tar utgangspunkt i vektor rom modellen hvor hvert dokument er representert som en vektor med en reell verdi. Vi skal her se på noen klassifiseringsmetoder hvor reelle tall kan brukes. Utgangspunktet for dette er contiguity hypoteses (Manning, Raghavan et al. 2008):

«Contiguity hypothesis: Documents in the same class form a contiguous region and regions of different classes do not overlap.»

For vårt tilfelle vil det si at dokumenter med stell av CVK vil skille seg fra dokumenter som ikke inneholder stell av CVK. Oppgaven i vektor rom klassifisering er å lage gode algoritmer som skiller klassene fra hverandre.

Læringsmodeller skiller også på om klassifiseringen kan skilles ved hjelp av en linje eller andre funksjoner. Hvis vi kan skille klassene fra hverandre ved hjelp av en linje omtales dette som en lineær klassifikator. Eksempler på en lineær klassifikator er Naive Bayes. Skillet mellom klassene blir da omtalt som decision boundary.

Under forhold der vi ikke kan skille klassene fra hverandre med en linje kan vi gjøre om funksjonen til å bli en annengradsfunksjon slik at man da kan skille klassene fra hverandre (Manning, Raghavan et al. 2008, side 303-306) .

4.6 Maskin læring

Maskin læring (ML) er et område innenfor kunstig intelligens hvor målet er å lage en algoritme/program for å lære av data. I 1959 definerte Arthur Samuel maskinlæring til følgende (Wikipedia):

«Field of study that gives computers the ability to learn without being explicitly programmed".

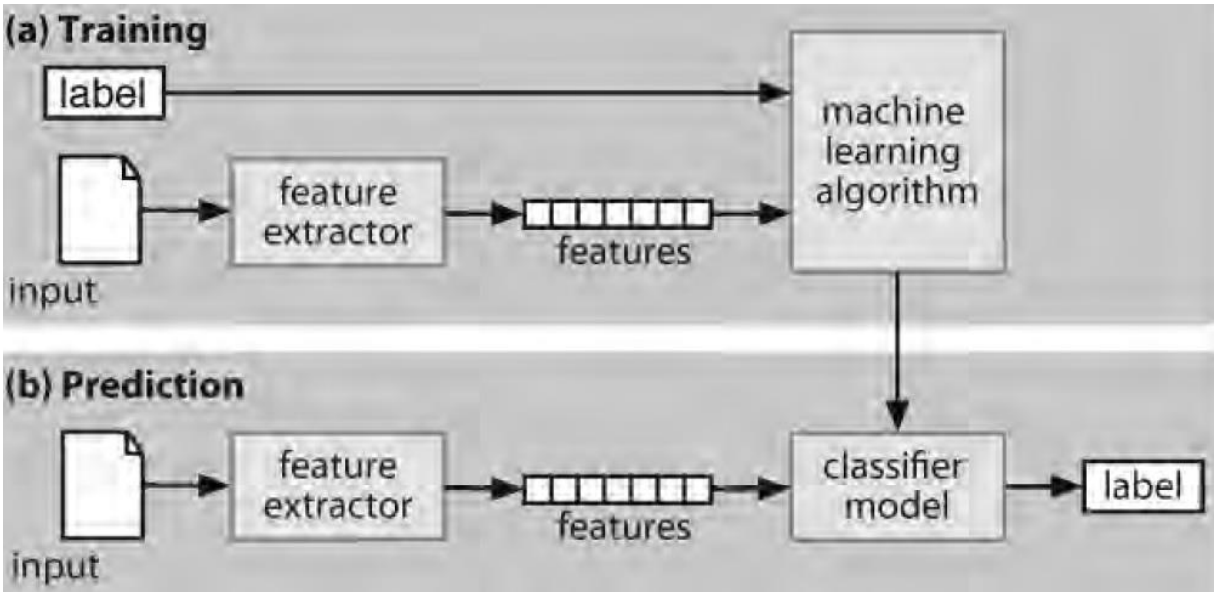
Det er to hoveddeler ML tar for seg, representasjon og generalisering. Representasjon av data tar for seg hvordan data bygges opp slik at data kan prosesserer videre av funksjoner som kan resonere på bakgrunn av disse. Etter at data har blitt prosessert av algoritmen, vil systemet ha lagd en generell funksjon som kan bli brukt på data systemet ikke har sett før. Hvor bra denne generaliseringen er avgjør hvor bra systemet fungerer i sin helhet.

Innen maskinlæring klassifiserer vi gjerne algoritmer i forhold til hva som blir gitt som inndata. Vi skal se på tre av disse her:

Supervised learning (Veiledet læring):

Inndata til slike algoritmer består som regel av problemstillingen og svaret til problemstillingen. Disse algoritmene forsøker å lage en generell modell fra inndata som brukes på usette data slik at modellen generaliserer så godt som mulig.

Figur 2 Rammeverket for en veiledet klassifisering.



Figur 2 viser hvordan treningen av en klassifikator fungerer hvor alle data er merket. I del a fores data inn til funksjonen som henter features vi vil se på. Features blir så presentert til maskinlæringsalgoritmen sammen med riktig merke av data. Ut fra algoritmen lages det en klassifikator modell. Denne modellen blir så brukt til å bestemme merke til data hvor merke er ukjent (Bird, Klein et al. 2009, side 222).

Unsupervised learning:

Dette er algoritmer som bruker data som ikke er merket på noen måte. Det vil si at man ikke kjenner resultatet basert på inndata. Målet her er å finne strukturer og klynger i data som kan bli brukt til å klassifisere ukjente data.

Semi-supervised learning:

En kombinasjon av veiledet og ikke veiledet læring blir kalt semi-supervised learning. Disse algoritmene bruker små mengder av annoterte tekster i kombinasjon med store data som ikke er annoterte.

Ved automatisk klassifisering brukes som regel supervised metoden, det vil si at vi har et datasett hvor vi kjenner merket til dokumentet. Vi kan da definere supervised maskinlæring som følger (Jurafsky):

Input: Ett sett dokumenter d

Klasser: et bestemt sett med klasser c ={$c_{1,}c_1$,…c_j}

Datasett med m merkede dokumenter, med hvert dokuments klasse. (d_1, c_1), (d_2, c_2), .. (d_m, c_m)

Utdata: en klassifikator $\gamma: d \rightarrow c$

4.6.1 Discriminative og generative klassifikatorer

Generative klassifikatorer lærer seg modeller ved å bruke felles sannsynlighet p(x,y), hvor x er input data og merke y. Ved å bruke Bayes regler for beregninger kan modellen forutsi sannsynligheten for p(y|x) , og på den måten velge den mest sannsynlige verdien for y. Eksempler på algoritmer i denne kategorien er Naive Bayes, Hidden Markov Models og n-gram modeller.

Disciminative klassifikatorer beregner sannsynlighet for p(y|x). Det vil si at modellen beregner sannsynligheten for et merke gitt input data(Ng and Jordan 2002). Eksempler på algoritmer i denne kategorien er logistisk regresjon, maximum entroy modeller og support vector machine.

4.6.2 Beslutningstrær

Et beslutningstre er en algoritme som ser ut som greiner på et tre, og skal velge rett merke for de data vi gir inn treet ved å gå ut på greinene til treet. Når vi kommer frem til løvet vil vi kunne sette rett merke på våre data. Treet består av beslutningsnoder, som ser om data består av de egenskapsverdier vi ser etter, og løv noder som gir verdier til inndata vi gir til algoritmen. I tillegg må vi bestemme desicion stump. En desicion stump er egentlig en løvnode for hver enkelt feature. Vi må da velge den feature som best skiller på merkingen vi er ute etter. Den letteste måten å gjøre dette på er å lage en desicion stump for alle våre egenskaper og beregne presisjon på hver enkelt feature. Vi velger da den som gir best presisjon som utgangpunkt for vårt beslutningstre. Det finnes også andre måter å finne ut hvilken egenskap som skal være utgangspunktet i treet. En annen metode er «information gain» som måler hvor nye bedre organisert input verdier blir ved å dele de opp gitt en feature. For å måle hvor uorganisert utgangpunktet av våre data er beregner vi «entropy» for deres merke. Verdien vil være stor hvis våre merker varierer mye, og lav hvis merkene ikke varierer mye. Entropi er definert som summen av sannsynlighet for hvert merke ganger logaritmen til sannsynligheten for hvert merke.

Når vi har beregnet entropy for hvert merke i data settet kan vi bestemme hvor mye mer organisert merkene blir hvis vi velger et annet utgangpunkt. Vi beregner så ny entropy hvis vi endre utgangspunktet. Vi trekker så fra den nye entropy fra den gamle. Vi velger da utgangpunkt i egenskapen som gir best information gain.

Beslutningstrær er enkle å forstå og hvor data er hierarkisk orientert. Ulempene er at beslutningstreet lett kan tilpasse(overfit) data når den kommer ned på løv nivå. Spesielt hvis det er et stort tre. En annen ulempe er at egenskapene må sjekkes i rekkefølge, selv om ikke egenskapene henger sammen(Bird, Klein et al. 2009, side 242-245).

4.6.3 Naive Bayes klassifikator

Naive Bayes er en generativ klassifikator og algoritmen har en statistisk tilnærming. Klassifikatoren antar at alle egenskaper er like viktige og at de er uavhengige av hverandre. Antagelsen om at egenskapene er uavhengige av hverandre er urealistisk, men gjør at vi kan beregne sannsynligheter.

Utgangspunktet er at vi har dokument d og klasse c (Dan Jurafsky and Manning):

$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$

Målet er å finne den største sannsynlighet for at et dokument tilhører en klasse c.

Hvis vi har 10 klasser og skal beregne sannsynligheten(P(d)) for dokumentene, vil denne sannsynligheten være en konstant og vi kan da skrive formelen som følger:

$P(c|d) = P(d|c)P(c)$, hvor P(d|c) omtales som likelihood og P(c) som prior

Sannsynligheten for P(d|c) er egentlig $P(x_1,x_2,\ldots x_n|c)P(c)$, hvor x er features til dokumentet. I vårt tilfelle vil dette være ord i dokumentet.

Naive Bayes har to forutsetninger:

1. Vi bruker bag of words for å representere features. Det vil si at vi da legger til grunn at rekkefølgen av ord ikke betyr noe.
2. Betinget uavhengighet.

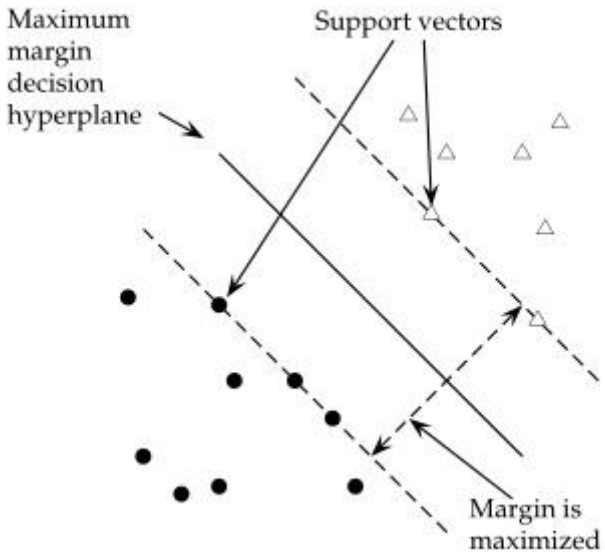
Antagelsene er som tidligere nevnt urealistiske, men gjør at vi da kan beregne sannsynlighet (lettere). Vi kan da gjøre beregningene med formelen: $C_{NB} = \text{argmax}_{c \varepsilon C} P(c_j) \Pi_{i \, \varepsilon \, posisjon} P(x|c)$

Ved å bruke Naive Bayes må vi være oppmerksom på at hvis et ord aldri har forekommet i treningsdata vil sannsynligheten i testdata for et nytt ord alltid være null. For å unngå slike tilfeller tar vi i bruk teknikken Laplace, også kjent som «add 1 smoothing». Det funksjonen gjør er å legge til 1 til hver count(f,label) verdi (Manning, Raghavan et al. 2008) .

4.6.4 Support vector machines (SVM)

Support vector machines er ansett for å være en god klassifikator. Målet til en SVM er å finne besluttningsgrensen mellom to klasser. Grensen skal i tillegg vært lengst mulig borte fra alle treningsdata. SVM er en metode som opererer i vektor rom, og er derav en vektor rom klassifikator.

Figur 3: Support vector machine. Bilde er hentet fra (Manning, Raghavan et al. 2008)



Figur 3 viser treningsdata til to klasser og hvordan SVM prøver å finne linjen som gir lengst avstand til alle treningsdata. Avstanden blir omtalt som margin til klassifikatoren.

4.7 Praktiske utfordringer ved tekst klassifisering

Det er ulike praktiske utfordringer når det kommer til klassifiseringsproblemer (Manning, Raghavan et al. 2008). Raghavan et al. mener at domene kunnskap kan ha større påvirkning på sluttresultatet enn å bytte maskinlæringsalgoritme. Hvor mye data, tilgang til nye data og om data er annotert betyr mye når vi skal velge maskinlæringsalgoritme.

Et annet moment som kan være viktig er feature engineering som går ut på å finne nye features som vil hjelpe algoritmen til å klassifisere bedre.

Hvis vi tar utgangspunkt i en e-post, kan vi velge å se mer nøye på emne. Manning, Raghavan et al. mener da at vi har to muligheter til å utnytte emne feltet i en e-post. Vi kan velge å vektlegge emnefeltet mer i korpus, eller vi kan ta emne ut og lage en egen klassifikator for emnene. I det siste eksemplet lager vi et eget feature space for et område i dokumentet.

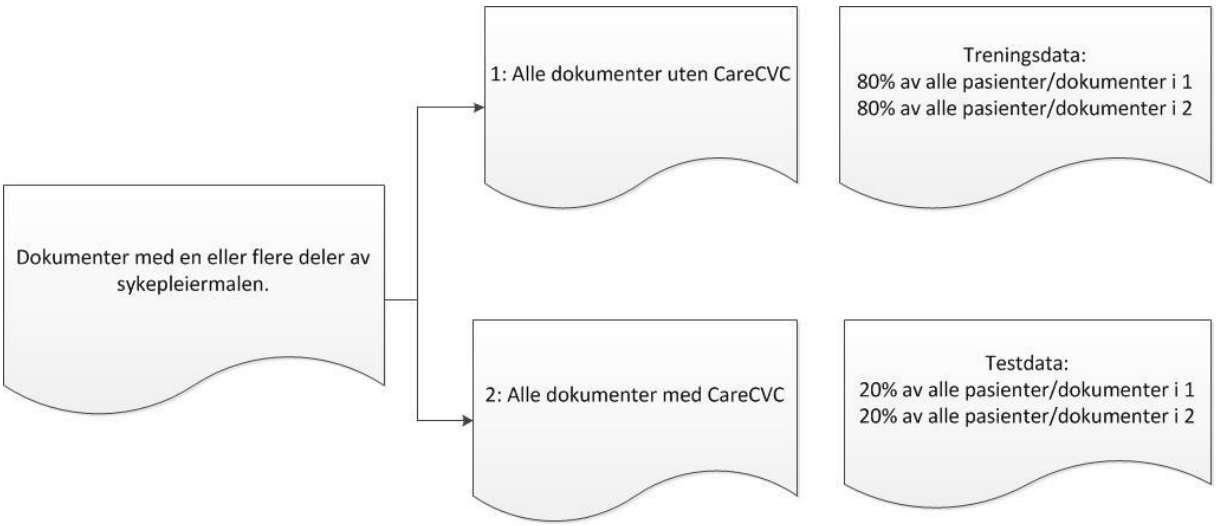
4.8 Fordeling av data

Da vi har et annotert materiale til rådighet vil det si at vi har en supervised klassifisering/learning modell å ta hensyn til. I delkapitler under vil vi se på ulike måter å dele inn data slik at vi kan lage en klassifikator. Vi må i det minste dele inn data i treningssett og testsett.

4.8.1 Trenings- og testdata

Figur 4 under viser hvordan vi kan dele inn data i trenings- og testdata.

Figur 4: Fordeling av trenings- og testdata



Figuren over viser at vi må finne alle dokumenter som har en eller flere deler av sykepleiermalen som er utgangspunktet for datamaterialet. I figurens del 1 plasseres alle dokumenter som ikke er merket med stell av CVK. I gruppe 2 er alle dokumenter som har blitt merket med stell av CVK. Treningsdata blir da populært med 80 % av data fra gruppe 1 (uten CareCVK), og 80 % av dokumentene fra del 1 (care CVC). På tilsvarende måte blir gruppen testdata fordelt, men da med en fordelingsnøkkel på 20 %. I realiteten vil det være de dokumenter som er igjen i del 1 og 2. For å plukke ut data fra delene 1 og 2 vil vi bruke stratifiserte utvalg som vil bidra til at jeg får en jevn fordeling av dokumenter med og uten stell av CVK i trenings- og test data (Wikipedia).

4.8.1 Kryssvalidering

Som tidligere nevnt må vi sette av en del av våre data til test data. Hvis testsettet er for lite kan evalueringen bli feil. Hvis vi øker test data, minker vi samtidig treningsdata noe som kan ha innvirkning på resultatene. En løsning på dette er å gjennomføre flere evalueringer på ulike testsett, for så å kombinere resultatene fra disse testene. Dette kalles kryssvalidering. Det vi gjør er å fordele hele korpus i N subset. Vi tester deretter modellen på det subsettet. Selv om hvert enkelt subset er for lite til å gi presise evalueringer, vil den samlede scoren bli basert på et stort sett med data og vil derfor bli pålitelig.

En annen viktig fordel med kryssvalidering er at vi kan undersøke resultatet for hvert sett og se på hvordan de ulike resultatene er. Hvis vi får lik score på alle subset kan vi med stor sikkerhet si at scoren er presis (Bird, Klein et al. 2009, side 241).

Svakheten med en slik modell er at vi trener og tester om hverandre med data.

4.8.2 trenings-, utviklings- og testdata

En annen måte å fordele data på er å bruke en 3 deling av korpus slik at feiltesting og tuning er skilt fra den endelige testen av modellen. Det er da vanlig at vi deler korpus inn i 3 deler: trenings-, utviklings- og testsett. Fordelingen mellom de ulike datasett varierer med mengden data vi har til rådighet, men fordelingen 60 % av data blir treningsdata, 20 % av data blir utviklingsdata og 20 % av data blir brukt til test av modellen, er en vanlig fordeling. Modellen trenes på data som er i treningssettet, vi sjekker feil og gjør forbedringer ved å bruke utviklingssetett og vi foretar endelig test av modellen med testsettet. Denne modellen er også veldig aktuell hvis vi skal teste flere algoritmer. Vi kan teste alle algoritmene på utviklingsdata, velge den modellen som gjør det best på utviklingsdata og teste denne på test data til slutt.

4.9 Evaluering

Evaluering av dokument klassifisering ble gjort ved hjelp av eksperimenter istedenfor analytisk. Grunnen til dette er at det ikke er mulig å spesifisere problemet systemet skal forsøke å løse (Sebastiani 2002). Isteden evalueres klassifiseringsoppgaver etter hvor effektive de er til å foreta de rette avgjørelser for klassifiseringen. Resten av dette delkapittel ser på to ulike måter å evaluere klassifikasjonssystemer på.

4.9.1 Evaluering ved hjelp av nøyaktighet (accuracy)

Den enkleste enheten å måle en modell på er å bruke nøyaktighet (accuracy). Nøyaktighet er et prosenttall på antall korrekte klassifiseringen modellen har gjort, delt på totalt antall klassifiseringer. Hvis vi tar utgangpunkt i figur 3 vil accuracy bli definert som følger:

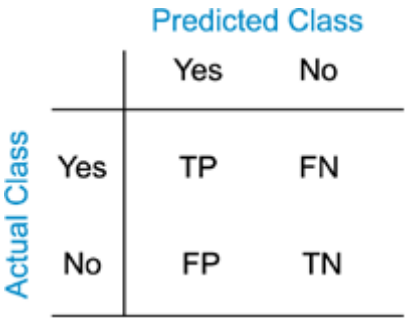
$$\frac{TP + TN}{TP + TN + FP + FN}$$

4.9.2 Evaluering basert på precision og recall

Nøyaktighet vil som regel være en god metode å evaluere en klassifikator på, med det er noen tilfeller der det vi søker å klassifisere er sterkt underrepresentert i korpus og vi må da kunne evaluere modellen vår på en annen måte. Det er her precision og recall kommer inn (Dan Jurafsky and Manning).

For å kunne beregne precision og recall er vi avhengig av en forvirringsmatrise (confusion matrix).

Figur 5: Forvirringsmatrise



Kilde: http://www.gepsoft.com/gepsoft/APS3KB/Chapter09/Section2/SS03.htm

Figuren over viser en Confusion matrix. På bakgrunn av gullstandard kan vi beregne en del tall som sier noe om resultatet av modellen som testes.

- True Positive (TP) er tester som virkelig er sanne og som modellen har funnet.
- True Negativ (TN) er svar som modellen ikke finner valid i forhold til det vi ser etter.
- False Positiv (FP) er svar som systemet har markert som en riktig verdi, men som i realiteten er feil.
- False negativ (FN) er verdier som systemet har tolket som feil, men som i virkeligheten er riktig.

På bakgrunn av denne matrisen kan vi beregne noen standardtall som blir brukt for å si noe om en test er god eller ikke.

Precision sier noe om hvor god modellen er til å finne enheter som vi definerte som relevant. Precision beregnes med følgende formel: $P = \frac{TP}{(TP+FP)}$

Recall sier noe om hvor god modellen er til å finne positive funn, eller hvilken del klassifiserte systemet riktig. Recall beregnes med følgende formel: $R = \frac{TP}{(TP+FN)}$

Hvis vi har flere ulike algoritmer som hver for seg har lagd en modell, men som gir ulike presisjon- og recall tall. Vi beregner da F-score, eller harmonisk gjennomsnitt.

F score beregnes som følger:

$F_1 = \frac{2PR}{(P+R)}$ (Bird, Klein et al. 2009, side 239).

5. Tilgjengelige resurser og verktøy

I dette kapittel beskrives programmer som er blitt brukt i denne oppgaven og ressurser som finnes tilgjengelig.

5.1 Mallet

Mallet er et program for statistisk naturlig språk prosessering, dokument klassifisering, clustering, topic modeling og informasjonsekstraksjon. Mallet har rutiner for å konvertere tekst til features og støtter blant annet maskinlæringsalgoritmer som Naive Bayes, Maximum entropy og beslutningstrær. Mallet har også støtte for å gjøre om tekst til numerisk representasjon ved å bruke pipes. En typisk pipe i Mallet vil da være tokenisering, fjerning av stopp ord og konvertering av sekvenser til vektor. Mallet er åpen kildekode og er utgitt under Common Puplic License (Mallet 2010).

En typisk pipeline for å behandle dokumenter på er hentet fra dokumentasjonen som finnes på Mallets sider[1]. Denne beskriver en måte å lage en pipe-line for behandling av dokumenter, og jeg vil gjennomgå denne i delene under.

5.1.1 Innlesning av dokumenter.

Før vi kan foreta klassifisering må vi lese inn dokumenter slik at vi kan konvertere dokumentene til feature vektorer. Det er mange måter og organisere dokumenter på, men den letteste måten er at et dokument er representert ved en fil. Videre er dokumentene organisert på en slik måte at dokumenter med ett merke (label) er lagt i en egen katalog. I denne oppgaven vil da dokumentene bli lagt i en av to kataloger, UtenCareCVC og CareCVC.

For å lage en liste over feature vektorer fra dokumenter, brukes kommandoen text2vectors i Mallet. Det er to parametere som er viktig å få med seg her: input og output. Input viser hvor data er hentet fra, og output viser hvor feature vektoren skal skrives til. Input parameteren peker på nivået over merket for dokumentene. Mallet går da inn i merkenes respektive kataloger og leser filene der.

Når Mallet lager indexer (med kommandoen text2vectors) i et dokument, foretar den i realiteten en tokenisering av dokumentet. Den tokeniserer alle ord og konverterer alle ord til små bokstaver. Det finnes flere opsjoner som kan brukes men det henvises da til hjelp funksjonen til kommandoen.

Text2vectors kan ta et komplett sett vektorer og splitte denne opp i trenings- og testsett. Det støttes også en måte å lage trenings- og testsett på som separate filer. Dette gjøres med å

bruke parameteren –use-pipe-from. På denne måten kan man generere en vektor fil for test og en for trening.

5.1.2 Klassifisering

Når filen for feature vektorer er lagd kan vi starte med å klassifisere dokumentene. Kommandoen vectors2classify brukes for å foreta trening av en klassifikator. Vi kan velge om kommandoen skal dele in trenings- og testdata eller vi kan lage to feature vektorer (en for test og en for trening). For å bruke to filer brukes parameterne –-training-file og –testning-file. Det er også mulig å angi hvordan rapporteringen skal foregå og hva man skal trene.

5.1.3 Klassifikatorer

Mallet har støtte for å kunne bruke ulike klassifikatorer. Ifølge dokumentasjonen støttes følgende algoritmer: NaiveBayes (default), Maximum Entropy, beslutningstrær og Winnow.

5.2 NLTK

Natural language toolkit (NLTK) er en plattform for å lage programmer som kan brukes på menneskelige språk. NLTK bruker python som programmeringsspråk. Det finnes over 50 korpa og leksikalske ressurser. Eksempler på tilgjengelige ressurser er wordnet, tokenisering og stemming. NLTK blir ofte brukt til undervisning når det kommer til programmering for områder som knyttes til naturlig språk. Hjemmesiden til NLTK er: www.nltk.org.

5.3 Qlikview

QlikView er et kommersielt produkt for business intelligence. QlikView blir gjerne brukt for å analysere data en bedrift samler inn (Abzaltynova and Williams 2013). I denne settingen blir QlikView brukt for å hente data til Evicare fra EPJ som brukes på A-hus. QlikView sammenstiller metadata og flytter/skriver filer til disk. Programmet er gratis å bruke for enkeltpersoner, men da med restriksjoner på flytting av program som er lagd. Det finnes egne brukerforum og det finnes mange ressurser på nettet. Hjemmesiden til QlikView er: http://www.qlik.com/.

5.4 Regulære uttrykk

Et regulært uttrykk (RE) er en sekvens av karakterer som sammen lager et mønster for søk. Relatert til å redigere et dokument er det funksjonen søk og erstatt (Wikipedia). Vi bruker RE for å søke etter forkortelser, feilstavinger og lignende i mange filer samtidig. I denne oppgaven har det blitt brukt til å sørge for at ord som CVK blir skrevet på lik måte i alle dokumenter.

6. Relatert arbeid

Det har ikke vært mulig å finne tidligere studier som går på dokumentklassifisering av sykepleienotater når det kommer til stell av CVK, eller generelt om dokumentklassifisering av sykepleiedokumenter. Jeg har søkt etter litteratur med en kombinasjon av følgende ord: document, classification, medical, domain, nurse, notes. Jeg har funnet noen studier som går på dokumentklassifisering i det medisinske domene, og vil omtale tre stykker her.

6.1 Identifying patien smoking status from medical discharge records

Artikkelen omtaler i2b2s utfordring om å kunne identifisere pasienters røykestatus basert på epikriser (Uzuner, Goldstein et al. 2008). Begrensningen i studien var å finne eksplisitt uttrykt røyke informasjon. Det vil si at informasjon som implisitt avslører røyking ble utelatt. Data ble avidentifisert, tokenisert, setninger ble identifisert, filene ble konvertert til xml format og data ble splittet i trenings- og testdata. I tillegg ble data annotert av flere lungeeksperter, slik at en gullstandard ble etablert. Ekspertene ble bedt om å klassifisere epikrisene i fem ulike røyke kategorier: Tidligere røyker, røyker, kjent røyker/ikke røyker hvor status ikke kommer frem, ikke røyker og ukjent. Korpa bestod av 502 avidentifiserte epikriser. Fordeling av data til trening var på 398 epikriser og 104 epikriser til test. De brukte presisjon, recall og f-score som metode. Utfordringen ble sendt ut og totalt 11 lag deltok i utfordringen. Lagene kunne hver for seg levere tre løsningsforslag. Det ble totalt levert inn 23 forslag. Forlagene løser problemet på mange ulike måter, alt fra regelbaserte løsninger til løsning som bruker maskinlæringsalgoritmer som SVM og beslutningstrær. I alt 12 av løsningsforslagene leverte F-score over 0,84.

6.2 MITRE system for clinical assertion status classification

I2B2 hadde i 2010 en utfordring på området «Challenges in natural language processing for clinical data». Oppgaven gikk ut på å klassifisere påstander assosiert med problemområder uthentet fra pasientjournaler. Clark et al. utviklet et system som kombinerte maskinlæring og regelbaserte teknikker for å finne negasjoner, spekulasjoner, hypotetiske og betinget informasjon (Clark, Aberdeen et al. 2011). Når det gjelder maskinlæring brukte de algoritmene for conditional random fields og maximum entropy. Den regelbaserte modulen bestod av regulære uttrykk, som var utviklet manuelt. De lagde videre et system som lette etter overskrifter i kliniske rapporter. Denne modulen var lagd ved å bruke regulære uttrykk, som markerer seksjonsgrenser. Ved funn av medisinske uttrykk skulle de kategorisere innenfor følgende grupper: Til stede, Ikke tilstede, mulig, hypotetisk, betinget og ikke assosiert med pasienten. De skulle gruppere følgende problemområder med gitte kategorier ovenfor: myocardial infarction og blobal hypoxic injury to the brain. Løsningen fungerte bra

med beste F-score på 0,9343. De konkluderer med at å bruke semantiske egenskaper om konsepter og informasjon om dokument struktur er bra som tillegg til regelbaserte og statistiske teknikker.

6.3 The Yale cTAKES extensions for document classification: architecture and application

I klinisk språkprosessering (KLP) annoteres syntaks og semantikk, men feature uthenting og representasjon av disse for å kunne utføre dokument klassifisering er utfordrende. Garla et al har i denne artikkelen adressert dette problemet, og vil forenkle feature uthenting og utviklingen av regel- og makinlæringsbaserte dokument klassifiseringssystemer (Garla, Re et al. 2011). De gjorde dette ved å videreutvikle cTAKES. De lagde et program basert på regulære uttrykk som skal finne NER og deteksjon av seksjoner i dokumenter. De tok i bruk siste versjon av NegEx (finne negasjoner) algoritmen og de utviklet en modul som lagrer annoteringer i en database. Det å lagre annoteringer i en database forenkler utviklingen av regelbaserte klassifikatorer. Ved å bruke SQL spørringer i databasen får vi enkelt frem vektorrepresentasjoner av dokumentene. Vektorene kan ved hjelp av SQL spørringer lett transformeres. Studien skal klassifisere radiologiske dokumenter basert på lever dekompensering. De fant videre ut at dokumentdelen en term forekom i er en viktig feature når det kommer til dokument klassifiseringen. De fant også ut at det å filtrere bort klinisk historie forbedret klassifikatorens effektivitet. For å oppnå best mulig resultat gjentok de følgende steg:

- Lagde annoteringer ved hjelp av YTEX
- Klassifiserte dokumenter ved regler
- Undersøkte dokumenter som ble klassifisert feil manuelt
- Rekonfigurerte YTEX for å forbedre feil
- Til sist ble feil de ikke kunne forbedre sent til domeneeksperter for evaluering.

De brukte maskinlæringsalgoritmene beslutningstrær, random trees og SVM.

De delte inn data i trenings- og testsett.

De trente algoritmene med tre typer data:

- Baseline: Data er annotert uten deres forbedringer
- Enkel: Bag of words representasjon, men uten dokumentseksjoner og negasjoner
- Rik: Alle features er med, også negasjoner og NER.

Som metode brukte de spesifisitet, presisjon, recall og F-score. De oppnådde gode resultater.

7. Design av eksperiment

I dette kapittel beskrives design på eksperimentet.

7.1 Kunnskap om stell av CVK i sykepleiedokumentasjonen

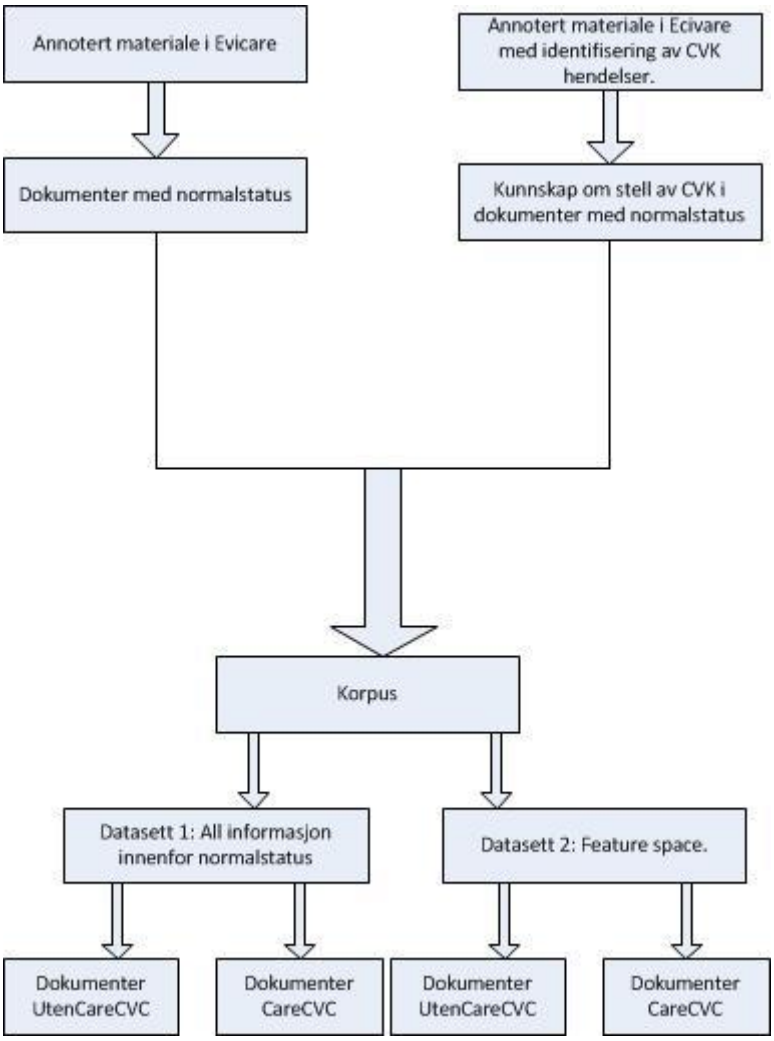
Det første som ble gjort var å skaffe kunnskap om Evicares annoterte materiale. Alle dokumenter som har betegnelsen CareCVC ble identifisert, og ved en manuell gjennomgang ble følgende identifisert om hvert enkelt dokument: om dokumentet inneholder normalstatus og eventuelt hvor CareCVC forekommer i normalstatusen. Vi vil da bruke det element fra normalstatus som omtaler CareCVC mest som inklusjonskriterie i studien.

7.2 Korpus

For å finne korpus til studien tar vi utgangspunkt i Evicares annoterte data. Første steg i denne prosessen var å identifisere dokumenter som innehar en eller flere deler av normalstatus. Neste steg var å finne alle dokumenter som inneholder data i delen som ble identifisert i punktet 7.1, som omtaler stell av CVK mest. Når dette var utført, var korpus for denne studien identifisert.

To datasett ble definert basert på korpus. Ett datasett vil hente ut all informasjon innenfor normalstatus. Det andre datasettet vil være et feature space, og kun inneha data fra den delen av malen som inneholder flest informasjon om stell av CVK. Hver av disse delene vil så bli splittet opp i to deler. En del hvor vi vet at CareCVC forekommer og en del hvor CareCVC ikke forekommer. Figuren under viser prosessen med å definere korpus og hvordan datasettene vil se ut.

Figur 6: Identifisering av korpa og fordeling av data



7.3 Feature engineering

Ved hjelp av feature engineering ble det undersøkt om klassifikasjonen kan bli bedre enn bare å dele inn datasett etter innhold i en del av dokumentet. Vi skal bruke funksjoner som fjerner forkortelser og stemming som feature engineering. Dette steget tar utgangspunkt i korpa som er definert i 7.2, Datasett 1 og 2. Ved å lage en frekvensordliste for hvert av datasettene letes det etter forkortelser. Hvis forkortelser finnes, lages det en regel ved å bruke regulære uttrykk som erstatter forkortelsen med teksten vi har bestemt. Denne prosessen ble gjentatt frem til vi sa oss fornøyd med endringene som er gjort.

Det samme utføres for stemming, men da på resultatet fra steg om forkortelser. Stemmingen ble gjort ved å bruke Snowball stemmeren som finnes som en egen ressurs i NLTK.

7.4 Kvalitetsmål

Kvalitetsmålene Presicion, Recall og F1-score ble brukt til å evaluere klassifikasjonen. Begrunnelsen for dette er at det er skeivfordeling av data i vårt materiale. Det vil si at det er mange færre dokumenter som inneholder stell av CVK enn de som ikke har det.

7.5 Fordeling av data

På bakgrunn av at datasettet er lite og at kryssvalidering bruker data flere ganger deler vi opp data settet i to deler. En del for trening, som består av 80 % av filene i datagrunnlaget, og en del til test, hvor resten av materialet havner. Det vil si at vi henter 80 % av filene med data i UtenCareCVC og 80 % av filene med data i CareCVC som danner treningsdata. På tilsvarende måte bygges testdata opp. Vi henter 20 % av filene med data fra CareCVC og 20 % av filene med data fra UtenCareCVC. Vi har i tillegg stratifisert dokumentene slik at data blir plukket ut vilkårlig.

7.6 Klassifisering

Som klassifikator skal vi bruke algoritmen Naive Bayes. Denne vil da bruke alle ordene som features og vi kan da sammenligne resultatene mot hverandre.

8. Eksperiment

Dette kapittel beskriver hvordan eksperimentet er løst.

8.1 Kunnskap om sykepleiedokumenter og stell av CVK

Vi bruker i denne delen Evicares annoteringsfiler som inneholder data for å skaffe oss kunnskap om sykepleiedokumentene. Det vil si at det er funnet hendelser om CVK i filen.

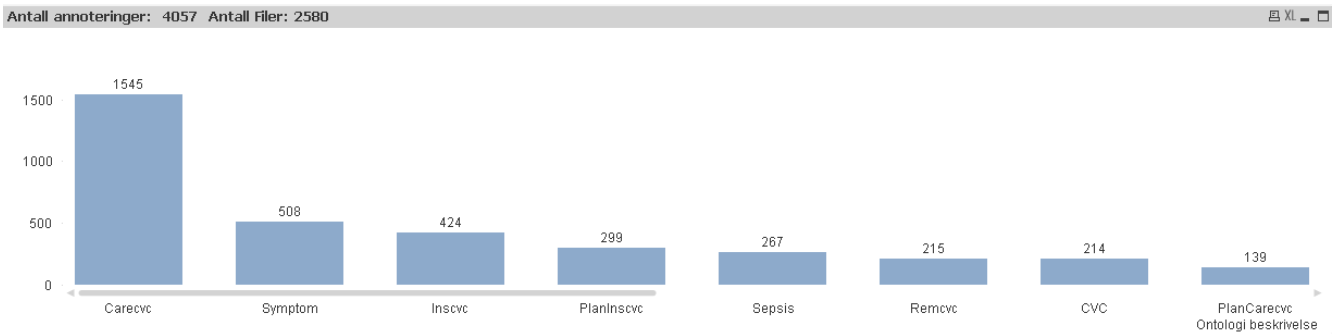
8.1.1 Steg 1: Innlesing av filer

Første steg i denne prosessen er å lese inn alle annoterte filer. Disse filene har jeg lagt i en egen katalog annoterte-CVK-funn. Programmet leser fra denne katalogen og skriver data til en egen qlikview fil i katalogen qvd. Vedlegg 5 viser koden som leser inn alla filer, splitter de i ulike dataelementer. Programmet tar 13 sekunder å kjøre.

8.1.2 Innlesing av annoterte filer

Neste steg i prosessen er å lese inn data fra qvd filen i steget over. Skriptet som gjør dette er vist i vedlegg 7 og er lagd i qlikview. Det tar 1 sekund å lese inn data. Av data kan vi få frem en del statistikker, som vist i figur 7.

Figur 7: Annoteringer



Figuren ovenfor viser at det er totalt annotert 4057 elementer i korpus, fordelt på 2580 filer. Av disse er det 1545 tags som er gjort på stell av CVK (Carecvc). Dette er de elementene vi er interessert i å studere videre. Ved å velge CareCVC vil vi kunne eksportere filnavnet og teksten til Excel for manuell registrering av data. Figur 8 viser et eksempel på eksporten som er gjort.

Figur 8: Eksport av data til kunnskapsinnhenting

2. Eksporter Data i tabellen til excel

Nr	Tag	desc	Tekst
1010590	T1	Carecvc	CVK er stelt da den lå helt åpen uten bandasjer
1024945	T1	Carecvc	CVK stell utført.
1024945	T1	Carecvc	Litt rød rundt suturene v/CVK.
1024945	T1	Carecvc	Tatt CVK-stell.
1024945	T1	Carecvc	Utført CVK stell
1024945	T2	Carecvc	Har CVK
1061801	T1	Carecvc	Cvk skift i dag 7/9-11
1076598	T1	Carecvc	Bandasje på CVK inngang er datert 3/6
1076598	T1	Carecvc	Bytt posiflow till CVK
1076598	T1	Carecvc	CVK-OK
1076598	T1	Carecvc	CVK-skift tatt
1076598	T1	Carecvc	CVK ble skiftet i dag
1076598	T1	Carecvc	CVK er trøg
1076598	T1	Carecvc	CVK fungerer godt
1076598	T1	Carecvc	CVK fungerte ikke på morgenen
1076598	T1	Carecvc	CVK går helt fint å skylle
1076598	T1	Carecvc	CVK går något bedre på kvelden

8.1.3 Innlesing av innsamlet kunnskap

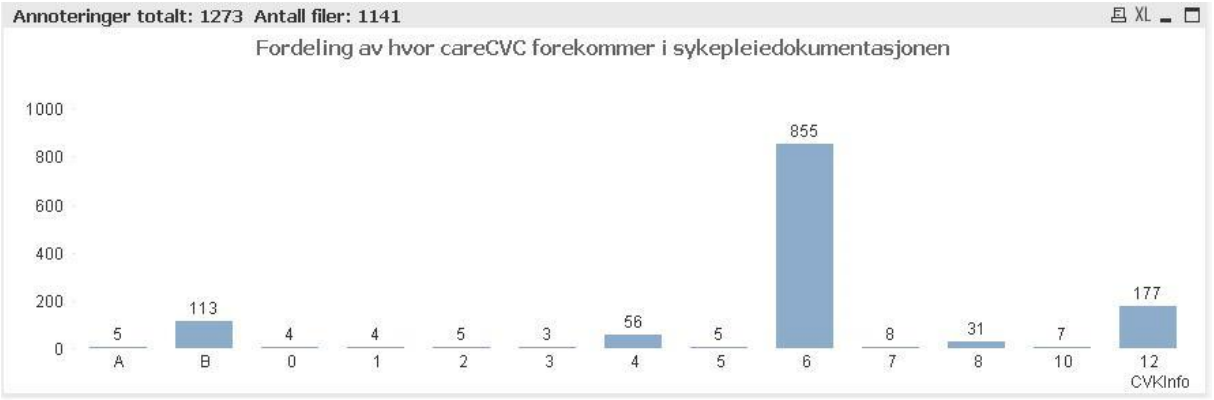
Dette steget tar for seg innlesning av innsamlet data etter manuell gjennomgang. Den manuelle gjennomgangen registrerte i hvilken del CareCVC finnes i hvert enkelt dokument (CVKInfo), og hvilke deler av normalsatus som eksisterer i dokumentet. Figur 9 viser hvordan data er registrert i Excel arket.

Figur 9: Resultat innsamling

F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
nr	Tekst	CVKInfo	IkkeAktuellt	Del1	Del2	Del3	A:	B:	Del4	Del5	Del6	Del7	Del8	Del9	Del10	Del11	Del12	Grunn
1	Skiftet fra treveiskraner til posiflow på CVK	12		0	0	1	1	1	1	1	1	1	0	0	0	0	1	
2	CVK	12		0	1	1	1	1	1	1	1	1	1	1	1	0	1	
3	Kommer med CVK	6		0	1	1	1	1	1	1	1	1	1	0	1	0	1	
4	Pas har CVK med 3-veiskraner på	6		0	0	0	0	0	1	1	1	0	0	0	0	0	1	
5	CVK ligger på samme sted	12		0	0	0	1	1	0	1	1	1	1	1	0	1	0	1	
6	Gjort CVK stell i dag.	6		0	0	0	1	1	0	1	1	1	1	1	0	1	0	1	
7	Det ble ikke utført CVK stell i går	6		0	0	0	1	1	0	1	1	1	1	1	0	1	0	1	
8	akutt dialysekateter hø		1															Ikke SPL mal
9	Fått CVK-skift	6		0	1	1	1	1	1	1	1	1	1	1	1	1	1	
10	hø side CDK		1															Ikke SPL mal

Dokumentet etter innsamling av data leses inn i QlikView for bearbeiding. Programmet som gjør dette vises i vedlegg 8. Programmet tar under 1 sekund å kjøre. Fordeling av hvilken del av sykepleiemalen annoteringen tilhører er vist i figur 10 under.

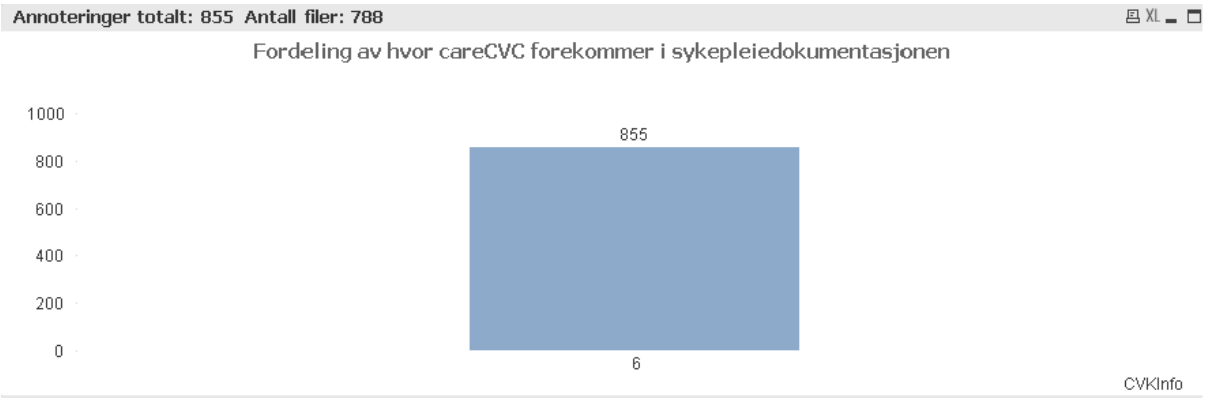
Figur 10: Fordeling av hvor stell av SVK forekommer



Av figur 10 ser vi at det er totalt 1273 annoteringer som er funnet innenfor normalstatusen. Annoteringene er fordelt på 1141 filer. Det vil si at det er noen filer som har flere annoteringer. Videre er det del 6 (Hud/vev/sår) fra normalstatusen som har mest data om CareCVC. Heretter refereres Del6 til innholdet dokumentene har i delen Hud/vev/sår.

Figuren 11 viser detaljer rundt del6. Det er da totalt 855 annoteringer som er gjort, fordelt på 788 filer. Listen av filer er eksportert ut til Excel for senere bruk (carecvc_dokumenter.xls).

Figur 11: Hud/vev/sår

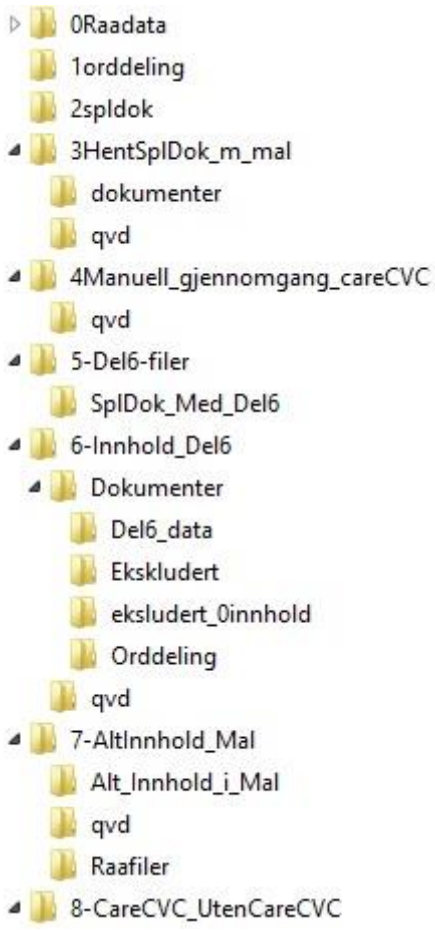


8.2 Korpus

Denne delen av oppgaven tar for seg arbeidet med å definere korpus for oppgaven.

Figuren under viser katalogstrukturen for å definere korpus.

Figur 12: Katalogstruktur



8.2.1 Dokumenter som inneholder normalstatus

Det er annotert 20795 dokumenter i Evicare, og disse er samlet i en egen katalog 0raadata. Vi skal i denne delen finne dokumenter som innehar en eller flere deler av normalstatus.

8.2.1.1 Steg 1: Ordsplitting

Første steg i prosessen er å skrive hvert ord på hver sin linje slik at vi senere kan lete etter avsnittene definert i normalstatus. Vedlegg 3 viser kildekoden for dette program som er

skrevet i python. Programmet henter et og et dokument fra katalogen 0raadata, erstatter mellomrom med ny linje og skriver filen ut til katalogen 1orddeling. Programmet tar 107 sekunder å kjøre.

8.2.1.2 Steg 2: Identifisering av dokumenter med normalstatus

Neste steg i prosessen er å finne dokumenter som inneholder en eller flere deler av normalstatus. Programmet leser en fil om gangen fra katalogen 1orddeling, leter etter overskriftene i sykepleiermalen og skriver ut overskriftene i egen fil i katalogen 2Spldok. Vedlegg 4 viser kildekoden for å finne disse dokumenter. Programmet tar 70 sekunder å kjøre.

Tabell 5: Listen under viser hvilke deler fil 22 inneholder av sykepleiemalen

Kunnskap/utvikling/psykisk
Åndedrett/Sirkulasjon
A:Åndedrett
B:Sirkulasjon
Ernæring/væske/elektrolyttbalanse
Eliminasjon
Hud/vev/sår
Annet/legedelegerte

8.2.1.3 Steg 3: Kopiere filer

Siste steg i prosessen er å kopiere filene med innhold over til en egen katalog. Dette gjøres i to steg:

1. Programmet leser inn filene fra katalogen 2spldok og skriver ut til katalogen 3HentSpldok_m_mal/qvd. Programmet er skrevet i qlikview og vises i vedlegg 5. Programmet tar 6 minutter å kjøre.
2. Siste steg i prosessen er å kopiere filene fra raadata katalogen til 3HentSpldok_m_mal/dokumenter. Programmet leser inn qvd-fil fra steget overfor, tar tak i filnavn som er registrert og kopierer filene fra raadata til 3HentSpldok_m_mal/dokumenter. Programmet er lagd i qlikview og vises i **vedlegg 6**. Programmet tar 8 minutter å kjøre. Vi har nå skilt ut dokumenter som inneholder sykepleie mal i en egen katalog.

8.2.2 Dokumenter som har Hud/vev/sår

Vi skal i denne delen finne dokumenter som har Del6 (Hud/vev/sår) i dokumentet. Det vil si at elementet fra normalstatus finnes i dokumentet. I steg 3 (del 1) identifiserte vi hvilke deler av sykepleiermalen hver enkelt fil har. Vi gjenbruker resultatet herfra til å hente ut alle filer som har Del6 i seg. Filene kopieres så til katalogen spldok_med_del6. I **vedlegg 9** er utskrift fra dette programmet, som tar 6 minutter å kjøre. Resultatet er at vi har identifisert 9251 filer som har med Del6 fra normalstatus.

8.2.3 Dokumenter med innhold i del 6

Neste steg er å skille ut dokumenter som har data i Del6 fra de som kun har malen representert i seg. Denne delen krever fem steg for å fullføres, og deloppgavene er beskrevet under.

8.2.3.1 Orddeling

Første steg i prosessen er å dele dokumentene slik at hvert ord kommer på en egen linje. Dette gjøres på samme måte som i del 8.1.1.1. Forskjellen her er at vi bruker kataloger for Steg 6-1 i vedlegg 3. Programmet er skrevet i Python og tar 48 sekunder å kjøre.

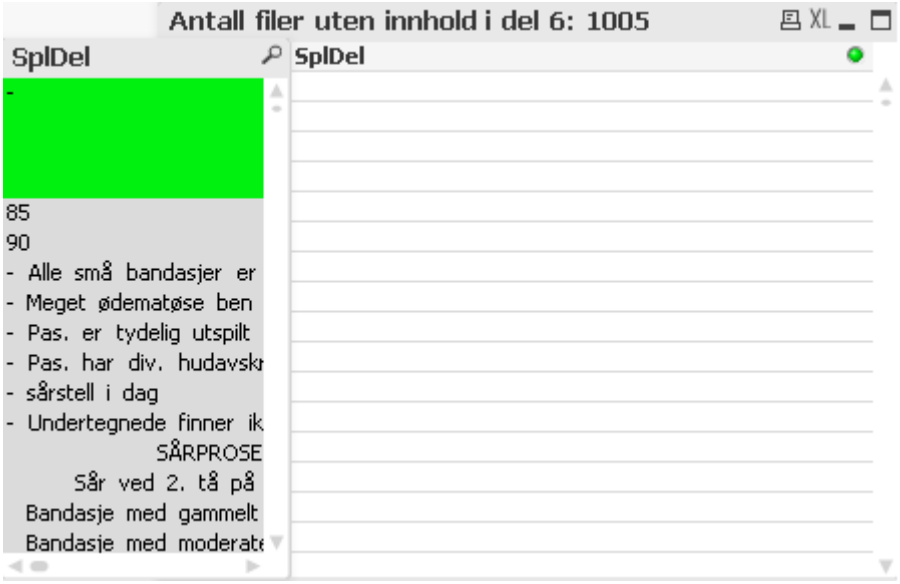
8.2.3.2 Innhold i Hud/vev/sår

Neste steg er å hente ut innholdet dokumentene har i Del6. Programmet for å gjøre dette er lagd i Python og tar 22 sekunder å kjøre. Programmet er vist som **vedlegg 10**. Programmet leser data fra katalogen orddeling og skriver data til Del6_data.

8.2.3.3 Identifisere filer med tomt innhold

I og med at dokumentene er skrevet med en mal er det slik at det ikke nødvendigvis data i Del6. Dette må undersøkes og fjernes da det ikke gir noen verdi å ha tomme dokumenter, eller dokumenter med kun mellomrom i seg. Bare ved å se på størrelse på filer i katalogen Del6_data finner vi 144 filer som ikke har noe innhold. Disse er fjernet fra katalogen og lagt til katalogen ekskludert. Gjennom vilkårlig sjekk av filene viser det seg at noen filer kun har mellomrom som innhold. Disse filene må også fjernes.

Figur 13: Tomt innhold i del Hud/vev/sår



Figur 13 viser at det 1005 dokumenter som ikke har noe innhold i Hud/vev/sår. Innholdet i filen er ett eller flere mellomrom, eller en bindestrek. Disse dokumentene kan da tas bort. Listen vist over, inkludert filnavn, er eksportert til Excel for videre arbeid. Programmet leser data fra katalogen Del6_data og skriver ut en fil i qvd katalogen. Programmet tar 2 minutter å kjøre, og ligger som vedlegg 4.

8.2.3.4 Fjerne tomme filer

Excel filen som ble eksportert leses inn i qlikview og filene flyttes fra katalogen Del6_data til ekskludert. Programmet ligger i vedlegg 11 og tar 27 sekunder å kjøre.

Vi har da 8219 filer i korpus. Innholdet av disse filene er kun data som er skrevet i delen Hud/vev/sår.

8.2.4 Hele innholdet i filene for korpus

Vi skal nå hente ut innholdet, innenfor normalstatus, for alle filer i korpus. Denne prosessen er delt inn i to deloppgaver, og blir beskrevet nærmere i sine respektive deler.

8.2.4.1 Klargjøring for å hente ut alt innhold i filene i korpus

Vi skal klargjøre data slik at vi kan hente ut alt innhold innenfor normalstatus. I kapittel 8.2.3.1 splittet vi ordene for alle filene i korpus. Vi skal gjenbruke disse data i denne delen, og kopiere filene til katalogen raadata i del 7. Programmet som gjør dette er lagd i qlikview og leser først inn alle filer i katalogen orddeling i kapittel 8.2.3.1. Programmet skriver videre filen ut til sitt eget internformalt, qvd. Programmet leser i neste omgang inn alle filer fra denne qvd filen og kopierer data fra orddeling til Raafiler. Programmet tar 5 minutter å kjøre, og utskrift av programmet ligger i vedlegg 12.

8.2.4.2 Uthenging av data fra filene i korpus

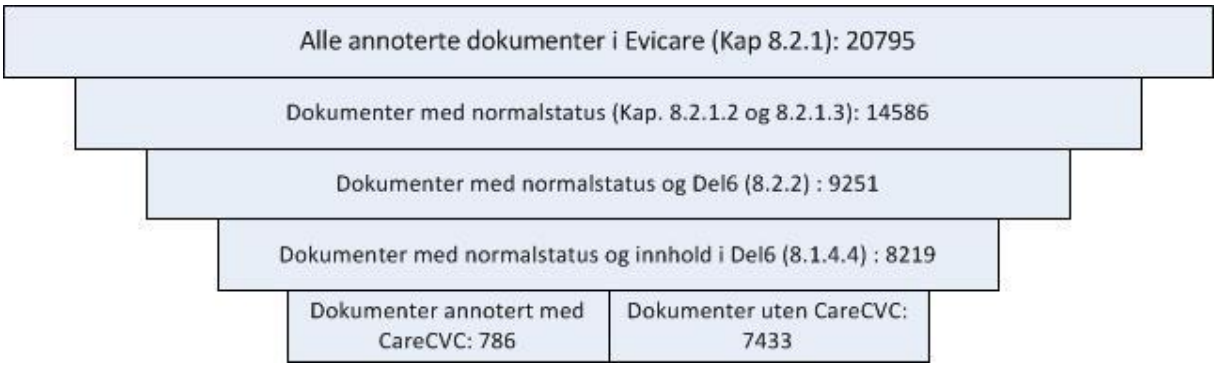
Programmet som skal hente ut data innenfor normalstatus er lagd i python og finnes i vedlegg 13. Programmet tar 50 sekunder å kjøre. Programmet leser data fra katalogen raafiler og skriver til katalogen Alt_innhold_i_mal. Utdata fra programmet er innholdet innenfor normalstatus, og selve overskriftene er tatt bort. Vi sitter nå med to typer datasett, som inneholder de samme filer og at de har data i Del6 (Hud/vev/sår), men hvor innholdet er todelt: Et datasett inneholder all tekst innenfor normalstatus, mens datasett 2 kun inneholder data som finnes innenfor Del6 /Hus/vev/sår).

8.2.5 Splitte korpa i CareCVC og UtenCareCVC deler

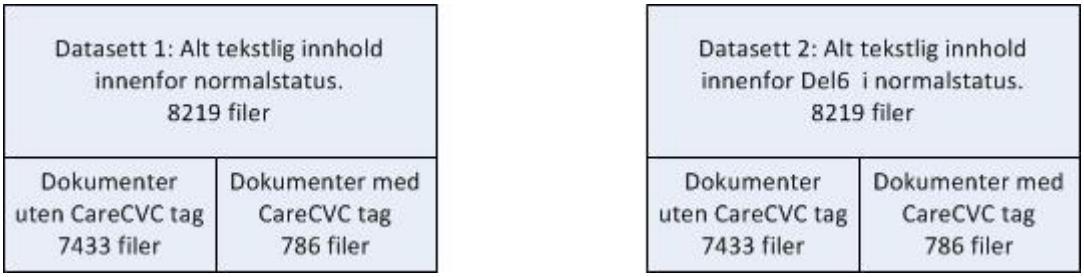
Siste steg i prosessen med å generere datasettene er å skille ut de dokumenter som er annotert med CareCVC. Dette gjøres ved å lese inn resultatet fra den manuelle gjennomgangen. Det er her filnavn som er avgjørende. Etter at filene er lest inn flyttes filene fra katalogen i steg 8.2.3.4 til katalogen dokumenter/CareCVC. Katalogen dokumenter/UtenCareCVC er resten av filene som ligger igjen i katalogen. Vi har da fordelt data i to grupper, en for CareCVC og en for UtenCareCVC. Fil fordelingen er da 786 og 7433 filer. Skriptet som gjør jobben er utviklet i qlikview og tar 23 sekunder å kjøre. Skriptet ligger som vedlegg 14.

En oppsummering av hvordan data i korpus er kommet frem er angitt i figuren under.

Figur 14: Filfordeling korpus



Figur 15: Datasett



Figuren over viser at vi har to datasett, Alt og Del6, dokumenter innenfor hvert datasett er de samme, men innholdet er ulikt.

8.3 Feature engineering

Vi skal se om feature engineering som utskriving av forkortelser og stemming har noen innvirkning på resultatet av dokumentklassifiseringen. Nedenfor vil vi se hvordan forkortelser og stemming er gjort.

8.3.1 Forkortelser

Det er lagd to programmer knyttet til denne prosessen. Ett program for å lage frekvensordliste og ett program for å endre forkortelser til ordets fulle skrivemåte.

Frekvensordliste:

Programmet lager en liste over filene som finnes i katalogen (CareCVC og UtenCareCVC). For hver enkelt fil leses filen inn, ordene splittes ved å bruke tokenize funksjonen til NLTK. Ordene blir lagt til en global variabel. Etter å ha lest inn alle filer og splittet opp ordene lages frekvensordlisten som i sin tur skrives ut til egen fil. Programmet er lagd i python. Ved å se på ordlisten manuelt kan forkortelser identifiseres. Utskrift av programmet ligger i vedlegg 15.

Endring av forkortelser:

På bakgrunn av de ord som er forkortelser legges ordene til i programmet om endrer ordene. Programmet tar for seg en og en fil, for hvert ord leter den etter forekomster av forkortelsen. Hvis forkortelsen finnes, endres ordet. Til slutt skrives filen ut i sin helhet til en egen katalog. Programmet er lagd i python og bruker regulære uttrykk for å gjøre endringene. Utskrift av programmet ligger i vedlegg 16. Denne jobben har vært repeterende for å finne forkortelser og er på ingen måte fullført da det ville tatt alt for lang tid. En oversikt over de ord som er endret finnes i vedlegget.

Som resultat av denne prosessen har vi nå 4 datasett. Ett for raadata og ett for forkortelser. Under hver av kategoriene finnes det en fordeling på UtenCareCVC og CareCVC.

8.2.2 Stemming

Vi skal her finne stammen av ordene i hvert enkelt dokument. Programmet leser inn en og en fil, ett og ett ord, sjekker stammen til ordet og skriver ut alle ordene til egen fil. Programmet bruker snowball stemmeren for norsk. Prosessen for å gjøre dette bruker tre programmer. Steg 1 er å splitte ett og ett ord på hver sin linje (Vedlegg 3). Steg 2 er å gjøre om alle ord til små bokstaver, dette ligger som vedlegg 17**.** Til slutt gjøres det for hvert ord oppslag mot snowball stemmeren. Hvis snowball finner ordet skrives ordet ut i sin endrede form. Finnes ikke ordet skrives ordet ut slik det er skrevet. Programmet er skrevet ut og ligger som vedlegg 18.

8.2.3 Oppsummering av frekvensordlister

Av tabellene under ser vi at antall ord går ned, i alle datasett, ved å bruke forkortelser og stemming. Den nederste tabellen viser tiden det har tatt å lage frekvensordlistene.

Tabell 6: Ordfordeling

	Antall ord		
	Originale filer	Etter Forkortelser	Etter Stemming
Del 6: CareCVC	2372	1896	1536
Del 6: UtenCareCVC	8743	6297	4926
Alt innenfor mal: CareCVC	9047	6541	5243
Alt innenfor mal: UtenCareCVC	38255	23333	17997

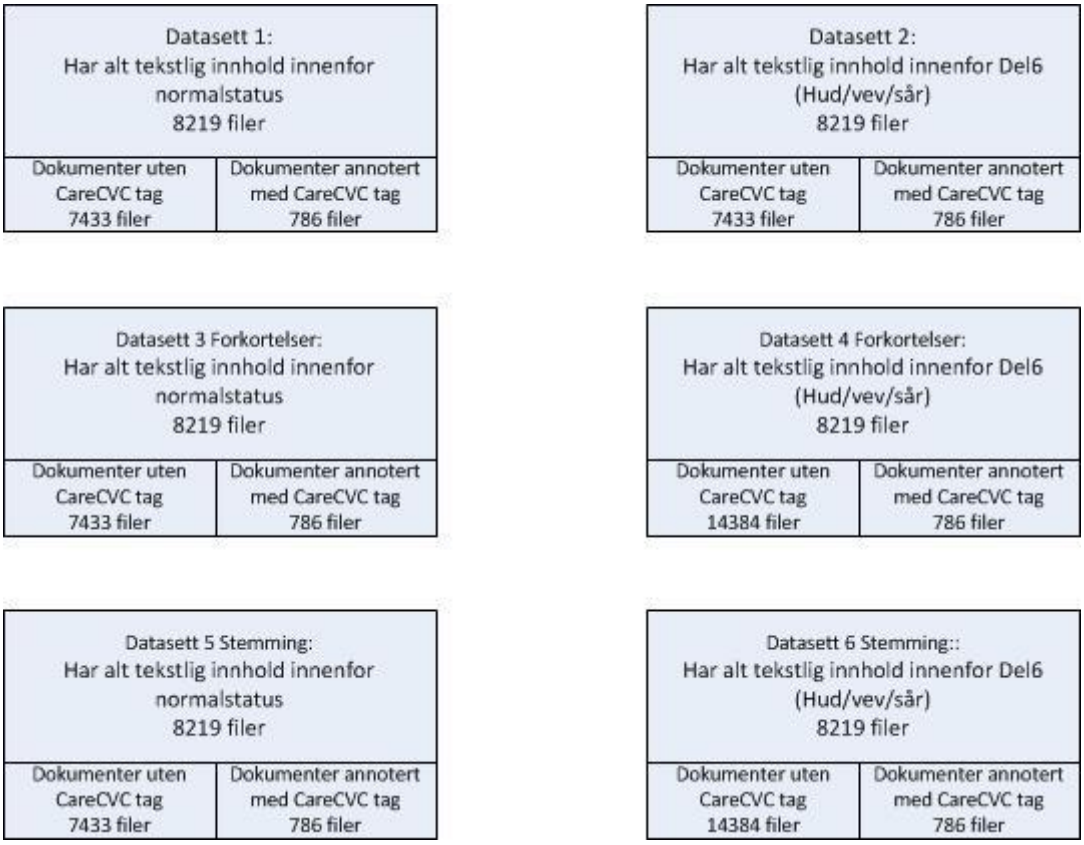
Tabell 7: Tidsbruk

	Tid for å kjøre i sekunder		
	Originale filer	Etter Forkortelser	Etter Stemming
Del 6: CareCVC	1	1	1
Del 6: UtenCareCVC	15	5	5
Alt innenfor mal: CareCVC	3	1	2
Alt innenfor mal: UtenCareCVC	53	37	37

8.2.4 Alle datasett

Vi har etter preprosesseringen totalt 6 datasett. Dokumentene er de samme i alle datasett, mens innholdet er ulikt. En oppsummering av alle datasett er vist i figur 15.

Figur 16: Alle datasett



8.4 Fordeling av data

Da vi skal klassifisere dokumenter på bakgrunn av all tekst innenfor normalstatus, mot kun Del6, vil vi kun bruke en læringsalgoritme. Vi har da ikke behov for å dele data inn i tre deler da vi ikke skal velge en algoritme som vi foretar den endelige testen på testdata. En gjengs fordeling av data blir da at treningsdata består av 80 % av alle data, både av dokumenter uten stell av CVK og de med stell av CVK. På tilsvarende måte fordeles resten av data til test. Vi

skal stratifisere data ved å eksportere data til to Excel filer. En for data med stell av CVK og en for data uten stell av CVK. Vi skal da fordele data på dokumentnivå. Vi eksporterer da kun over filnavn. Ved å bruke vilkårlig funksjonen i Excel genererer vi et vilkårlig tall mellom 0 og 1 til hvert dokument. Tallene blir kopiert, og limt inn i en ny rad. Da ved og kun lime inn tall, og ikke formel. Dette er viktig da tallene vil endre seg hvis funksjonen fortsatt er med. Vi sorterer verdiene i synkende rekkefølge. Fordelingen blir da som følger: de to første elementene tilfaller trening, tredje tilfaller test, de fire neste tilfaller trening, åttende element tilfaller test, før de to siste tilfaller trening. Denne prosessen gjentas til vi er tomme for filer, i både delene med og uten CareCVC.

Prosessen for denne oppgaven er delt i fire deler:

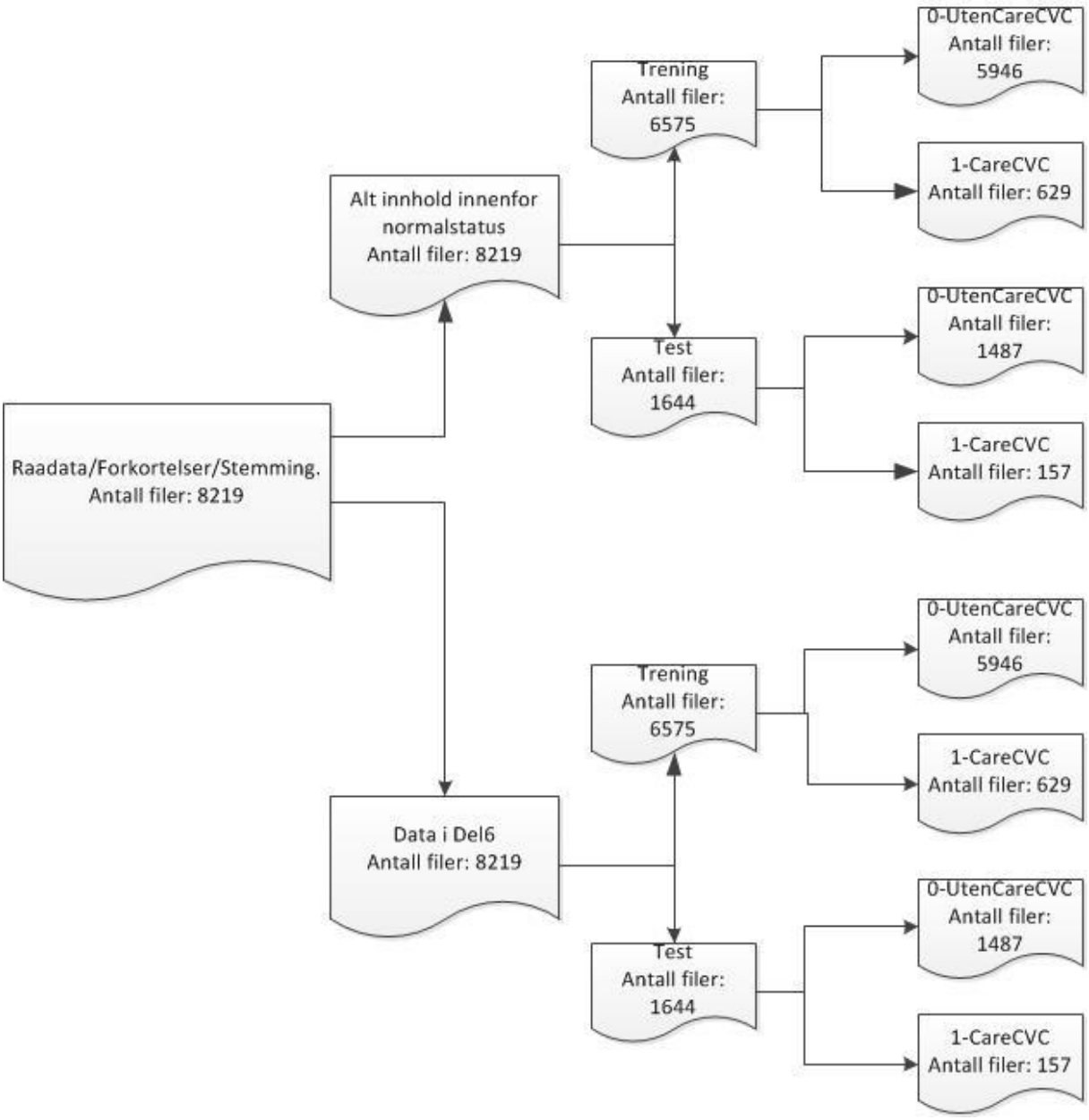
1. Først leses alle filer som er kategorisert med stell av CVK. Filnavn eksporteres til Excel. Det samme gjøres for filer UtenCareCVC.
2. Det lages vilkårlige tall for hver fil. Tallene kopieres til egen kolonne, og det sorteres i synkende rekkefølge. Tildeling av gruppe skjer som tidligere anvist. Tabellen under viser utskrift av dette. Det samme gjøres for den andre gruppen
3. Fordelingsfilene blir lest inn, og data blir fordelt til sine respektive kataloger. Det skilles da på katalogene 0-UtenCareCVC og 1-CareCVC. Dette er gjort da Mallet krever at data er satt sammen på denne måten.
4. Programmet som leser inn filnavn for data med og uten av CVK, og som fordeler filene er lagd i qlikview. Utskrift av programmet ligger i vedlegg 19.

Tabell 8: Fordeling av filer i trenings- og testsett

ID	Random	Fordeling
1	0.999939	Trening
2	0.9985657	Trening
3	0.9967651	Test
4	0.9961243	Trening
5	0.9956665	Trening
6	0.9951477	Trening
7	0.9951172	Trening
8	0.9947205	Test
9	0.9947205	Trening
10	0.9938965	Trening
11	0.9922791	Trening
12	0.990387	Trening
13	0.9900818	Test
14	0.9891052	Trening
15	0.9880371	Trening
16	0.9853821	Trening
17	0.9853516	Trening
18	0.9847717	Test
19	0.9776306	Trening
20	0.9748535	Trening
21	0.9747009	Trening

Tabellen over viser hvordan data er delt inn i trenings- og testdata. Fordeling vil bli gjeldende for alle deler av data vi har. Se figuren under for mer detaljer om hvordan data totalt sett ser ut.

Figur 17: Total fil fordeling



Figuren over viser hvordan data er fordelt. Første nivå er raadata (filer som har data i Del6 i normalstatus). Denne fordelingen gjelder for alle datasett vi har, viser til figur 15 for oversikt over disse..

Neste steg er fordelingen av data basert på hvilke deler innen normalstatusen som er med. Alt eller kun data i del6. Deretter er hver av disse bolkene delt inn i trenings- og testdata, som igjen er delt inn i data med 0-UtenCareCVC og 1-CareCVC.

8.5 Klassifisering

Data er nå fordelt slik at vi har tre ulike hovedgrupper, som hver for seg er delt inn i to undergrupper, som igjen er delt inn i trenings- og testdata, som vist i figur 16. Vi kan velge og ikke foreta inndelingen på måten som er beskrevet, men heller la Mallet gjøre dette for oss. Her er dokumentene skilt på trening og test da vi ikke kan være sikker på at Mallet inndeler trenings- og test data på samme måte. Vi skal da lage feature vektorer basert på pipe-line funksjonaliteten Mallet støtter, og vi skal bruke algoritmen Naive Bayes for klassifiseringen. Resten av kapitelet tar for seg hvordan generering av feature vektorer for treningsdata og testdata er foretatt.

8.5.1 Raadata

Dette kapittelet vil ta for seg generering av feature vektorer for data som er uten endringer fra opprinnelig dokument. Det er skilt på Alt (all tekst innenfor malen sykepleiere bruker), og Del6 (kun data innenfor del6 av malen).

8.5.1.1 Alt

Vi starter med å generere feature vektorer for treningsdata og testdata. For treningsdata gjøres dette med kommandoen :

bin/text2vectors --input data/1-Raadata/Alt/Training/* --output resultater/RaadataAlt/AltTraining.vectors

For testdata gjøres dette med kommandoen:

bin/text2vectors --use-pipe-from resultater/RaadataAlt/AltTraining.vectors --input data/1-Raadata/Alt/Test/* --output resultater/RaadataAlt/AltTest.vectors

Vi ser av kommandoene at data hentes fra ulike kataloger, men at resultafilene skrives til samme katalog.

For å trene og teste klassifikatoren gjøres dette med følgende kommando:

bin/vectors2classify --training-file resultater/RaadataAlt/AltTraining.vectors --testing-file resultater/RaadataAlt/AltTest.vectors > resultater/RaadataAlt/result.txt

Resultatet av treningen er vist i figuren under:

Tabell 9: Resultat av trening for rådata med Alt innenfor malen

		Klassifikasjon	
		0	1
Gullstandard	0-UtenCareCVC	1554	33
	1-CareCVC	122	35

8.4.1.2 Del6 (Hud/vev/sår)

På samme måte som i steget over lages det egne feature vektorer for trening og test. Test settet genereres da på samme måte med å bruke pipe funksjonen i Mallet. Vektorene samles i en egen katalog. Til slutt kjører treningen på disse vektorene. Utskrift av kommandoer for å lage vektorene, samt treningen følger vedlagt. Til slutt vises resultatet fra treningen.

Treningsdata:

bin/text2vectors --input data/1-Raadata/Del6/Training/* --output resultater/RaadataDel6/Del6Training.vectors

Testdata:

bin/text2vectors --use-pipe-from resultater/RaadataDel6/Del6Training.vectors --input data/1-Raadata/Del6/Test/* --output resultater/RaadataDel6/Del6Test.vectors

Trening og test av klassifikator:

bin/vectors2classify --training-file resultater/RaadataDel6/Del6Training.vectors --testing-file resultater/RaadataDel6/Del6Test.vectors > resultater/RaadataDel6/result.txt

Tabell 10: Resultat for trening av rådata, kun del6

		Klassifikasjon	
		0	1
Gullstandard	0-UtenCareCVC	1441	46
	1-CareCVC	71	86

8.4.2 Forkortelser

Vi skal her se på kommandoer og resultat for data som har blitt utsatt for endring av utvalgte forkortelser i dokumentene.

8.4.2.1 Alt innhold innenfor sykepleiemalen

Trenings feature vektorer genereres med kommandoen:

bin/text2vectors --input data/2-forkortelser/Alt/Training/* --output resultater/ForkAlt/ForkTraining.vectors

Test feature vektorer genereres med kommandoen:

bin/text2vectors --use-pipe-from resultater/ForkAlt/ForkTraining.vectors --input data/2-forkortelser/Alt/Test/* --output resultater/ForkAlt/ForkTest.vectors

Treningen av klassifikatoren gjøres med kommandoen:

bin/vectors2classify --training-file resultater/ForkAlt/ForkTraining.vectors --testing-file resultater/ForkAlt/ForkTest.vectors > resultater/ForkAlt/result.txt

Tabell 11: Resultater for trening av forkortelser, Alt

		Klassifikasjon	
		0	1
Gullstandard	0-UtenCareCVC	1455	32
	1-CareCVC	124	33

8.4.2.2 Innhold i del6 av sykepleiemalen

Feature vektorer for treningssettet genereres av kommandoen:

bin/text2vectors --input data/2-forkortelser/Del6/Training/* --output resultater/ForkDel6/ForkTraining.vectors

Feature vektorer for testdata lages med kommandoen:

bin/text2vectors --use-pipe-from resultater/ForkDel6/ForkTraining.vectors --input data/2-forkortelser/Del6/Test/* --output resultater/ForkDel6/Del6Test.vectors

Trening av klassifikatoren gjøres med kommandoen:

bin/vectors2classify --training-file resultater/ForkDel6/ForkTraining.vectors --testing-file

resultater/ForkDel6/Del6Test.vectors > resultater/ForkDel6/result.txt

Tabell 12: Resultatet for trening av forkortelser, del6

		Klassifikasjon	
		0	1
Gullstandard	0-UtenCareCVC	1437	50
	1-CareCVC	70	87

8.4.3 Stemming

Her er resultatene og kommandoene for å generere feature vektorer for trening og test data, samt resultatet etter at stemming er foretatt på data fra forkortelser.

8.4.3.1 Alt innhold innenfor sykepleiemalen

Generering av feature vektorer for treningsdata:

bin/text2vectors --input data/3-stemming/Alt/Training/* --output resultater/StemmingAlt/StemTraining.vectors

Generering av feature vektorer for testdata:

bin/text2vectors --use-pipe-from resultater/StemmingAlt/StemTraining.vectors --input data/3-stemming/Alt/Test/* --output resultater/StemmingAlt/StemTest.vectors

Trening av klassifikator:

bin/vectors2classify --training-file resultater/StemmingAlt/StemTraining.vectors --testing-file resultater/StemmingAlt/StemTest.vectors > resultater/StemmingAlt/result.txt

Tabell 13: Resultat for trening av stemming, alt

		Klassifikasjon	
		0	1
Gullstandard	0-UtenCareCVC	1447	40
	1-CareCVC	114	43

8.4.3.2 Datainnhold I del6 av sykepleiemalen

Kommando for feature vector for treningsdata:

bin/text2vectors --input data/3-stemming/Del6/Training/* --output resultater/StemmingDel6/Training.vectors

Kommando for feature vektorer for testdata:

bin/text2vectors --use-pipe-from resultater/StemmingDel6/Training.vectors --input data/3-stemming/Del6/Test/* --output resultater/StemmingDel6/Test.vectors

Kommander for klassifikator trening og resultat av trening:

bin/vectors2classify --training-file resultater/StemmingDel6/Training.vectors --testing-file resultater/StemmingDel6/Test.vectors > resultater/StemmingDel6/result.txt

Tabell 14: Resultat for trening av stemming, del 6

		Klassifikasjon	
		0	1
Gullstandard	0-UtenCareCVC	1438	49
	1-CareCVC	63	94

9. Resultater

På bakgrunn av resultatene i del 8.5 er det beregnet presicion, recall og F1-score for alle deler av analysene. Resultatet for del ulike delene er angitt i tabellen under.

Tabell 15: Kvalitetstall for klassifiseringen av testdata

Beskrivelse	Precision	Recall	F_1-score
Rådata med alt innhold innenfor normalstatus	51,47 %	22,29 %	31,11 %
Egenskapsområde for Rådata	65,15 %	54,78 %	59,52 %
Forkortelser med alt innhold innenfor normalstatus	50,77 %	21,02 %	29,73 %
Egenskapsområde for Forkortelser	63,50 %	55,41 %	59,18 %
Stemming med alt innhold innenfor normalstatus	51,81 %	27,39 %	35,83 %
Egenskapsområde for Stemming	65,73 %	59,87 %	62,67 %
Maksimal forbedring	14,26 %	37,58 %	31,56 %

Forbedringer internt i hvert enkelt datasett er vist i tabell 8. Dette er differansen mellom å klassifisere egenskapsområdet mot rådata, i hvert enkelt tilfelle.

Tabell 16: Forbedring av kvalitetstall intern i gruppene

	Precision	Recall	F_1-score
Rådata	13,68 %	32,49 %	28,41 %
Forkortelser	12,73 %	34,39 %	29,45 %
Stemming	13,92 %	32,48 %	26,84 %

For å se forbedret resultat mellom de ulike preprosesseringssteg som er gjort er det regnet ut differanse mellom egenskapsområdet og rådata i første kolonne, deretter er differansen mellom egenskapsområdene regnet ut. Tabell 9 viser disse tallene

Tabell 17: Forbedring av kvalitetstall for preprosessering

	Precision	Recall	F_1-score
Rådata	13,68 %	32,49 %	28,41 %
Forkortelser	-1,65 %	0,63 %	-0,34 %
Stemming	0,58 %	5,09 %	3,15 %

10. Evaluering og diskusjon

Analyse av resultatet:

Generelt kan vi se at resultatene for klassifiseringen, uavhengig av preprosessering, ikke er spesielt gode. Av tabell 7 ser vi at beste score for precision er 65,8 %, recall 59,9 % og F_1-score 62,7 %. Resultatet mellom å klassifisere all data uten endringer mot preprosessering av egenskapsområde, forkortelser og stemming, viser en stor forbedring. Her øker precision med 14 %, recall med 38,5 % og F_1-score 31,6 %.

Av første rad i tabell 9 ser vi en god forbedring mellom rådata og egenskapsområdet. Vi får ikke store forbedringer av å foreta stemming og utskrivning av forkortelser. Recall øker med litt over 5 % som den beste forbedringen. Det mest interessante er at utskrivning av forkortelser ikke forbedrer klassifiseringen i det hele tatt for precision og F1-score.

Det at utskrivning av forkortelser gjør at resultatet for precision og F_1-score faller er veldig overraskende. Fra tabell 4 ser vi at antall ord i vokabularet minker ved å foreta utskriving av forkortelser i forhold til rådata. Antall ord burde komme frem som en forbedring av klassifiseringen. Årsaken til hvorfor forbedringen ikke skjer er vanskelig å svare på. En svakhet ved frekvensordlisten fra tabell 4 er at denne burde vært generert på datasettnivå. Det er to forklaringer jeg ser som årsak til at resultatet minker:

1. Det er mulig er at det er tilført feil i materialet som følge av ekspandering av forkortelser.
2. Når dataprogrammet Mallet genererer egenskapsvektorer tokeniserer programmet teksten selv, og det er mulig det innføres endringer med teksten her.

Generelt er ikke resultatet bra for klassifiseringen, men det er en stor forbedring av å se på et egenskapsområde i forhold til hele teksten når det kommer til å finne forekomster av stell av CVK. Som preprosessering av data er det brukt tokenisering, utskrivning av forkortelser ved bruk av regulære uttrykk, frekvensordlister, normalisering av tokens og stemming, Ved å studere frekvensordlistene kommer det frem en del forkortelser av medisinsk karakter som ikke er skrevet ut da forståelsen av disse ikke er på plass. Det er også mange sammensatte ord som ikke er skilt, og det er mange ord som ikke er norske (svenske, danske og engelske). Dette vil nødvendigvis får følger for stemmingen, som igjen påvirker resultatet til slutt.

Preprosesseringsteknikker som ikke er blitt brukt i studien er stavekontroll, NER og fjerning av stoppord. NER ble vurdert til ikke å gi noe mer informasjon til denne oppgaven da sykepleienotater skrives i avidentifisert form. Det vil si at navn, adresser og lignende ikke skal forekomme i dokumentene. Stavekontroll ble vurdert til å bli for omfattende da det finnes mange språk i dokumentene.

Det er ikke foretatt stemming med utgangpunkt i rådata. Dette burde ha blitt gjort da det kunne vist om stemming er avhengig av forkortelsene eller ikke. Klassifiseringen kunne vært gjort med flere algoritmer. Da datasettet er lite, og anbefalingene er å bruke en enkel

klassifisering, ble enkel klassifisering gjort. Det kunne uansett vært interessent å se om andre algoritmer ville gitt bedre resultater på det samme datamaterialet.

Hvor er hendelsen stell av SVK dokumentert og hva mister vi ved kun å bruke egenskapsområde?

Gjennom å studere det annoterte materialet viser det seg at stell av SVK er annotert i 1356 filer, hvor 1141 filer innehar en eller flere deler av sykepleiemalen, og av disse er 788 filer annotert i Del6 (Hud/vev/sår). Det vil si at stell av SVK forekommer i 84 % av det annoterte materialet i sykepleiedokumenter, og at neste 70 % av disse finnes i Del6 i malen. Dette tyder på at vi godt kan bruke Del6 av sykepleiemalen som et utgangspunkt for å identifisere pasienter som har hatt SVK i løpet av sitt sykehusopphold. Da denne masteroppgaven har sett på filnivå vet vi ikke om vi mister noen pasienter som har hatt SVK i løpet av sitt sykehusopphold ved å bruke Del6 som utgangspunkt for klassifiseringen. Jeg vil mene at dette uansett er bedre enn resultatet ved å lete i den administrative delen av EPJ, der vi kan forvente å finne 10 % av pasientene som har hatt SVK.

Programvare som er brukt:

NLTK:

NLTK er en programpakke som har eksistert i mange år og som er veldokumentert. I tillegg bruker NLTK Python som programmeringsspråk, noe som gjør at vi kan finne hjelp fra begge områder. Programmet er gratis å bruke.

QlikView:

QlikView er et kommersielt produkt, men som kan brukes gratis av enkeltpersoner. Utfordringen er å dele programmer som er utviklet. Det er mulig å importere skript inn i en ny personlig lisens av programmet, men hver enkelt må da lage grensesnittet på nytt. Det er en stor fordel at det finnes mange brukere av QlikView og de har en egen portal for brukere som er gratis.

Mallet:

Mallet har vært ganske enkelt å ta i bruk. Utfordringen er at det er lite dokumentert, og at det ikke er lett å finne hjelp på nettet (McCallum 2012). Det er en del kommandoer du kan bruke i Mallet etter hvilket format dine data er på. Det er vanskelig å få oversikt over de ulike funksjonene, noe som gjør at du ikke vet når de ulike funksjonene skal brukes. I tillegg er det noen funksjoner ikke fungerer, samt at kryssvalideringen ikke gir output som gjorde meg skeptisk til Mallets prosess for kryssvalidering. Dette var en av grunnene til at jeg ikke valgte å dele inn data i kryssvalidering, og heller delte inn test- og treningsdata selv.

Hva kan dette brukes til?

Som tidligere nevnt er det å finne pasienter som har CVK vanskelig ved å bruke data fra en strukturert journal. Denne oppgaven kan være til hjelp for å finne pasienter som har CVK på et eller annet tidspunkt i sykehusoppholdet, ved å se om dokumentene omhandler stell av CVK. For sykehuset er det viktig å kunne finne flest mulig av pasientene som har SVK. Dette gir at vi må fokusere på recall fra tabell 7, da recall sier noe om andelen av riktig klassifiserte forekomster i forhold til gullstandarden. Beste recall i denne studien er på 59,9 %, etter at forkortelser er skrevet ut og stemming er foretatt. Resultatet er dog mye bedre enn om vi bruker all tekst innenfor sykepleiermalen.

Vi kan også se at det å bruke prosesser fra NLP ser ut til å bidra til at klassifikatoren blir bedre, selv om bidraget er lite. Da denne oppgaven ikke har hatt mulighet til å se etter alle forkortelser, og få kjørt stemming så effektivt som mulig, er det absolutt forbedringspotensialer ved å jobbe mer med disse prosessene.

Hva har jeg lært?

Jeg har lært veldig mye om et fagområde jeg visste veldig lite om fra før av. Min småirritasjon over at Google viser meg linker til sider som ikke er relevant for meg, blir nok mindre kommentert fra nå av. I tillegg har jeg fått jobbe tettere i et forskningsprosjekt som har vært interessant og utfordrende. Det har vært en til tider frustrerende prosess, men til slutt har dette vært en positiv erfaring å ta med seg videre.

11. Konklusjon

Denne masteroppgaven har sett på hvor stell av SVK forekommer i sykepleiedokumenter, med utgangspunkt i det annoterte materialet til Evicare. Av totalt 1356 filer hvor stell av SVK er markert, er 84 % av disse dokumenter et sykepleiedokument. Videre har vi sett at av disse sykepleiedokumentene er 69 % av forekomsten av stell av SVK knyttet til Del6 av dokumentmalen. Vi har ved dette vist at stell av SVK forekommer oftest i sykepleiemalens Del6(Hud/vev/sår).

Klassifikatoren som er blitt lagd gjennom denne masteroppgaven viser at det er en stor gevinst ved å bruke deler av sykepleiemalen i forhold til alt innhold innenfor malen. Ved kun å bruke data som ikke er manipulert på noen måte, øker precision med 13,7 %, recall med 32,5% og F_1-score med 28,4 %.

Når vi ser på resultatene for å ekspandere forkortelser viser ikke denne noen forbedringer for klassifiseringen. Her går resultatene for precision og F_1-score ned.

For klassifisering av data hvor ekspandering av forkortelser er foretatt først, for deretter å foreta stemming øker resultatene litt. Precision er nesten uforandret, mens recall øker med 5 % og F_1-score med litt over 3 %.

12. Videre arbeid

Det å få tak i mer data, og mer annoterte data, mener jeg er en god start for å forbedre klassifisering av sykepleiedokumenter når det kommer til stell av CVK.

Jeg ser av arbeidet at det med preprosessering som stemming og fjerning av forkortelser bør sees mer grundig på, da dette ikke er fullkomment i denne oppgaven. I tillegg bør dokumentene gjennomgå stavekontroll som en del av preprosesseringen.

Et annet program enn Mallet burde brukes til klassifiseringen da programmet er lite dokumentert. Jeg tror at det for en dreven person i faget vil Mallet være midt i blinken.

Det bør også vurderes å gjøre klassifiseringen med flere læringsalgoritmer da dette kan bedre resultatene.

Fra denne masteroppgaven har vi sett at Del6 i sykepleiemalen inneholder nesten 60 % (788/1356) av de annoterte data for stell av CVK. Det ville vært interessant å se om hvor mange pasienter som glipper som følge av å bruke Del6 for å lete etter pasienter med SVK i løpet av sitt sykehusopphold. Hvis dette tallet er lavt kan vi bruke Del6 som en første identifikator på å identifisere pasienter med SVK, eller bruke resultatet herfra i kombinasjon av de andre annoteringsmerkene.

Bibliografi

Abzaltynova, Z. and J. Williams (2013). "Developments in Business Intelligence Software." Journal of Intelligence Studies in Business **3**(2).

Bird, S., et al. (2009). Natural language processing with Python, O'Reilly Media, Inc.

Bruce, L.-E. (2012). "Processing Electronic Medical Records: Ontology-Driven Information Extraction and Structuring in the Clinical Domain."

Cho, P. S., et al. (2003). Automatic section segmentation of medical reports. AMIA Annual Symposium Proceedings, American Medical Informatics Association.

Clark, C., et al. (2011). "MITRE system for clinical assertion status classification." Journal of the American Medical Informatics Association **18**(5): 563-567.

Coden, A. R., et al. (2005). "Domain-specific language models and lexicons for tagging." Journal of biomedical informatics **38**(6): 422-430.

Coiera, E. (2003). Guide to Health Informatics.

Dan Jurafsky and C. Manning. "Naive Bayes." from https://class.coursera.org/nlp/lecture/25.

Dan Jurafsky and C. Manning "Precision, Recall and F measure."

Garla, V., et al. (2011). "The Yale cTAKES extensions for document classification: architecture and application." Journal of the American Medical Informatics Association **18**(5): 614-620.

Jurafsky, D. "Text Classification." from https://class.coursera.org/nlp/lecture/36.

Liu, H., et al. (2001). "Disambiguating ambiguous biomedical terms in biomedical narrative text: an unsupervised method." Journal of biomedical informatics **34**(4): 249-261.

Mallet, M. A. (2010). a machine learning for language toolkit, 2002.

Manning, C. D., et al. (2008). Introduction to information retrieval, Cambridge university press Cambridge.

McCallum, A. K. (2012). "Review of MALLET." from http://journalofdigitalhumanities.org/2-1/review-mallet-by-ian-milligan-and-shawn-graham/.

Meystre, S. M., et al. (2008). "Extracting information from textual documents in the electronic health record: a review of recent research." Yearb Med Inform **35**: 128-144.

Nadeau, D. and S. Sekine (2007). "A survey of named entity recognition and classification." Lingvisticae Investigationes **30**(1): 3-26.

Ng, A. Y. and M. I. Jordan (2002). "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes." Advances in neural information processing systems **2**: 841-848.

O'Grady, N. P., et al. (2002). "Guidelines for the prevention of intravascular catheter–related infections." Clinical infectious diseases **35**(11): 1281-1307.

Pasientsikkerhetsprogrammet (2014). "Forebygging av infeksjon ved sentralt venekateter." from http://www.pasientsikkerhetsprogrammet.no/no/I+trygge+hender/Innsatsomr%C3%A5der/Forebygging+av+infeksjon+ved+sentralt+venekateter.14.cms.

Penz, J. F., et al. (2007). "Automated identification of adverse events related to central venous catheters." Journal of biomedical informatics **40**(2): 174-182.

Ruch, P., et al. (2003). "Using lexical disambiguation and named-entity recognition to improve spelling correction in the electronic patient record." Artificial intelligence in medicine **29**(1): 169-184.

Sebastiani, F. (2002). "Machine learning in automated text categorization." ACM computing surveys (CSUR) **34**(1): 1-47.

Sørli, A. B. (2007). "Bruk av elektronisk dokumentasjon i sykepleietjenesten: Papir eller elektronisk dokumentasjon..... gjør det noen forskjell?".

Uzuner, Ö., et al. (2008). "Identifying patient smoking status from medical discharge records." Journal of the American Medical Informatics Association **15**(1): 14-24.

Wikipedia. "Bag of words model." from http://en.wikipedia.org/wiki/Bag-of-words_model.

Wikipedia. "Document classification." from http://en.wikipedia.org/wiki/Document_classification.

Wikipedia. "Natural language processing." from http://en.wikipedia.org/wiki/Natural_language_processing.

Wikipedia. "Regular Expression." from http://en.wikipedia.org/wiki/Regular_expression.

Wikipedia. "Stratified sampling." from http://en.wikipedia.org/wiki/Stratified_sampling.

Wikipedia, M., learning. "Macine learning." from http://en.wikipedia.org/wiki/Machine_learning.

Xu, H., et al. (2007). A study of abbreviations in clinical notes. AMIA Annual Symposium Proceedings, American Medical Informatics Association.

Vedlegg

Vedlegg 1: SQL spørring SVK

Koden under viser prosedyrekoden for innlegging av sentralt venekateter. Jeg brukte deretter koden til å finne ut hvor mange pasienter som har fått operert inn CVK.

```
SQL> select Kodeid, kode, tekst from dwmedisinskekoder where kodeid = 1114569;

    KODEID KODE                    TEKST
-------------------------------------------------------------------------------
   1114569 PYGC00                  Innlegging av sentralt venekateter


SQL> Select count(*) from DWMEDTJENESTE where tjenestekode = 1114569;

  COUNT(*)
----------
       198
```

Vedlegg 2: Taushetserklæring

Taushetserklæring **IDI, NTNU**
for forskere, studenter og andre med tilgang til sensitiv informasjon **01.12.20012**

NTNU kan opptre både som databehandlingsansvarlig og forskningsinstitusjon relatert til bruk og oppbevaring av sensitiv informasjon, både om enkeltpersoner og andre forhold. Innsamling, forvaltning og bruk av slik informasjon er underlagt spesielle prosjekter, konsesjoner, tillatelser, samtykke eller avtaler som skal være kjent av de involverte. Som regel skal arbeid eller forskning være søkt og godkjent av en regional komité for medisinsk og helsefaglig forskningsetikk[1] eller av Norsk Samfunnsvitenskapelig Datatjeneste som er universitetets personvernombud[2]. Det er din plikt å sette deg inn i de reglene som gjelder for deg og ditt prosjekt. NTNU har egne vegledere på nett for helsefaglig forskning[3] og annen forskning på sensitiv informasjon[4]. Denne erklæringen om taushet er hjemlet i følgende lover:

Universitets- og høyskoleloven, § 4-6. Studentenes taushetsplikt:
En student som i studiesammenheng får kjennskap til noens personlige forhold, har taushetsplikt etter de regler som gjelder for yrkesutøvere på vedkommende livsområde. Institusjonen skal utarbeide taushetsplikterklæring som må underskrives av de studenter dette er aktuelt for.

Lov om helsepersonell:
Helsepersonell skal hindre at andre får adgang eller kjennskap til opplysninger om folks legems- eller sykdomsforhold eller andre personlige forhold som de får vite om i egenskap av å være helsepersonell. (§21)
Helsepersonell er personell med autorisasjon eller lisens, personell i helsetjenesten eller i apotek som yter helsehjelp, elever og studenter som yter helsehjelp. (§ 3)
Samarbeidende personell og personer som bistår med elektronisk bearbeiding av taushetsbelagte opplysninger har samme taushetsplikt som helsepersonell. (§ 25)
Taushetsplikten gjelder tilsvarende for personell i pasientadministrasjon. (§ 26)

Forvaltningsloven, § 13:
Enhver som utfører tjeneste eller arbeid for et forvaltningsorgan, plikter å hindre at andre får adgang eller kjennskap til det han i forbindelse med tjenesten eller arbeidet får vite om:

- noens personlige forhold, eller
- tekniske innretninger og fremgangsmåter samt drifts- eller forretningsforhold som det vil være av konkurransemessig betydning å hemmeligholde av hensyn til den som opplysningen angår.

Spesialisthelsetjenesteloven, § 6-1:
Enhver som utfører tjeneste eller arbeid for helseinstitusjon som omfattes av denne loven, har taushetsplikt etter forvaltningsloven §§ 13 til 13e. Taushetsplikten gjelder også pasientens fødested, fødselsdato, personnummer, statsborgerforhold, sivilstand, yrke, bopel og arbeidssted. Opplysning om pasientens oppholdssted kan likevel gis når det er klart at det ikke vil skade tilliten til helseinstitusjonen.

Straffeloven, § 121:
Den som forsettlig eller grovt uaktsomt krenker taushetsplikt som i henhold til lovbestemmelse eller gyldig instruks følger av hans tjeneste eller arbeid for statlig eller kommunalt organ, straffes med bøter eller med fengsel inntil 6 måneder. Begår han taushetsbrudd i den hensikt å tilvende seg eller andre en uberettiget vinning eller utnytter han i slik hensikt på annen måte opplysninger som er belagt med taushetsplikt, kan fengsel inntil 3 år anvendes. Denne bestemmelse rammer også taushetsbrudd m.m. etter at vedkommende har avsluttet tjenesten eller arbeidet.

Taushetsplikten **gjelder til enhver tid** (også i fritid, etter at arbeidsforholdet er opphørt og lignende). Taushetsplikten gjelder i utgangspunktet **overfor alle andre, med unntak av personer underlagt taushetsplikt og som har tilgang til akkurat de samme opplysningene som deg i samme prosjekt**. I samsvar med dette erklærer jeg å forplikte meg til taushet om alt jeg i stillings medfør får vite om noens private forhold. Taushetsplikten gjelder ikke bare utad, men også overfor andre ansatte og tillitsvalgte for hvem saken/forholdet må anses uvedkommende. Opplysninger som jeg iht. mitt arbeidsforhold plikter å holde min overordnede orientert om, omfattes ikke av taushetsplikten. Det samme gjelder opplysninger som jeg etter andre lover eller rettens kjennelse er pålagt å gi.

Hvem 14/12-12 ..... Sted Dato Signatur

Underskrevet avtale leveres administrasjon på IDI for arkivering i ePhorte. Kopi til vegleder el. prosjektleder.

[1] helseforskning.etikkom.no
[2] www.nsd.uib.no/personvern/
[3] www.ntnu.no/dmf/helseforskning
[4] www.ntnu.no/studier/phd/personopplysninger

Vedlegg 3: Orddeling

```
# -*- coding: utf-8 -*-

#Kopier filen til C:\Python27\Lib

#starter filen med å skrive from Setninger import*

import time

start_time = time.time()

import nltk.data

import string

import os


#Kataloger for del 8.1.1.1

InnData = 'E:\Master_data\\Data\\0Raadata'

UtData = 'E:\Master_data\\Data\\1orddeling'


#Kataloger for del 8.1.4.1 i dokumentet

InnData = 'E:\Master_data\Data\\5Innhold_del6\\Dokumenter\\SplDok_med_malinnhold'

UtData = 'E:\Master_data\\Data\\6-orddeling\\UtenCareCVC'


#Kataloger for del 8.2.2

#For å håndtere Del6_carecvc

#InnData = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Kun_Del6\\CareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\orddeling\\Kun_Del6\\CareCVC'


#For å håndtere Del6_Utencarecvc

#InnData = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Kun_Del6\\UtenCareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\orddeling\\Kun_Del6\\UtenCareCVC'


#For å håndtere Alt_carecvc

#InnData = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Alt_innhold\\CareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\orddeling\\Alt_innhold\\CareCVC'
```

```
#For å håndtere Alt_Utencarecvc

InnData = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Alt_innhold\\UtenCareCVC'

UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\orddeling\\Alt_innhold\\UtenCareCVC'



#henter filer i in katalog

def hent_filnavn(sti):

    liste = os.listdir(InnData)

    return liste



filliste = hent_filnavn(InnData)

AntFiler = len(filliste)

x = 0

while x < AntFiler:

    fil = InnData + "\\" + filliste[x]

    clean = open(fil).read().replace(' ','\n')

    clean = clean.strip()

    ut_fil = UtData + "\\" + filliste[x]

    testfil = open(ut_fil,'w')

    testfil.write(clean)

    testfil.close()

    x+=1

print time.time() - start_time, "Sekunder"
```

Vedlegg 4: Spldok

```
# -*- coding: utf-8 -*-
#Kopier filen til C:\Python27\Lib
#starter filen med å skrive from Setninger import*
import time
start_time = time.time()
import nltk.data
import string
import os
from nltk.tokenize import word_tokenize
tokenizer = nltk.data.load('tokenizers/punkt/norwegian.pickle')

#Kataloger for del 8.1.1.2 i oppgaven
InnData = 'E:\Master_data\Data\\1orddeling'
UtData = 'E:\Master_data\Data\\2spldok'

def hent_filnavn(sti):
    liste = os.listdir(InnData)
    return liste

filliste = hent_filnavn(InnData)
AntFiler = len(filliste)
x = 0
AlleOrd = []

while x < AntFiler:
    fil = InnData + "\\" + filliste[x]
    ut_fil = UtData + "\\" + filliste[x]
    with open(fil, 'rb') as fp:
        para = fp.readlines()
    AntSet = len(para)
    y=0
    while y < AntSet:
        clean = para[y]
```

```
        setning = clean.replace('\r\n','')

        if (setning.startswith('Kommunikasjon/sanser') or
setning.startswith('Kunnskap/utvikling/psykisk') or
setning.startswith('Åndedrett/Sirkulasjon') or setning.startswith('A:Åndedrett') or
setning.startswith('B:Sirkulasjon') or
setning.startswith('Ernæring/væske/elektrolyttbalanse') or
setning.startswith('Eliminasjon') or setning.startswith('Hud/vev/sår') or
setning.startswith('Aktivitet/funksjons-status') or
setning.startswith('Smerte/søvn/hvile/velvære') or
setning.startswith('Seksualitet/reproduksjon') or
setning.startswith('Sosialt/planlegging') or
setning.startswith('Åndelig/kulturelt/livsstil') or
setning.startswith('Annet/legedelegerte')):

            ut = open(ut_fil,'a')

            ut.write(setning+'\r')

            ut.close()

        y += 1

    x+=1

print time.time() - start_time, "Sekunder"
```

Vedlegg 5: Last Data

```
///$tab Main
SET ThousandSep=' ';
SET DecimalSep=',';
SET MoneyThousandSep=' ';
SET MoneyDecimalSep=',';
SET MoneyFormat='kr # ##0,00;kr -# ##0,00';
SET TimeFormat='hh:mm:ss';
SET DateFormat='DD.MM.YYYY';
SET TimestampFormat='DD.MM.YYYY hh:mm:ss[.fff]';
SET MonthNames='jan;feb;mar;apr;mai;jun;jul;aug;sep;okt;nov;des';
SET DayNames='ma;ti;on;to;fr;lø;sø';

//SPL Dokumenter som har en eller flere deler fra SPL malen i seg
//let vSteg3 = 'E:\Master_data\Data\2spldok\';
///$tab Oversikt filer
//Lager en liste over filene som har mal fra spl dokumenter i seg
sub DoDir(Root)
	let FoldNo = mid(Root,12, 10 ); //returns 'cd'. //SubField(Root,'\',2);
	FOR Each Ext in 'txt' //filtype å lete etter
		FOR Each File in FileList(Root & '*.' & Ext)
			Files:
			load '$(File)' as Name,
			FileTime('$(File)') as FileTime,
			RangeSum(Peek('FileCount'),1) as FileCount
			AutoGenerate 1;
		NEXT File
	NEXT Ext
	For Each Dir in DirList (Root & '*')    //leter i underkataloger
		Call DoDir(Dir)
	NEXT Dir
end sub
///$tab Les inn filer
//Leser inn data for alle filer i hver katalog
```

```
sub LesFiler

	let vRader = NoOfRows('Files')-1; //Finner antall filer som er lest

	for i=0 to vRader

		let vFilNavn = peek('Name',i,'Files'); //Henter filnavn


		//Henter Tidsstempel fra filnavn

		let vFilTid_tmp = right('$(vFilNavn)',23); //henter ut siste del av
filen. tidsstempel + .txt (dd-mm-åååå-tt-mm-ss.txt)

		let vFilDato_tmp2 = left('$(vFilTid_tmp)',10);  //henter ut dd-mm-ååå

		let vFilDato = Replace('$(vFilDato_tmp2)','-','.'); //dato på format
dd.mm.åååå


		let vFilTid_tmp2 = right('$(vFilNavn)',12);          //henter ut tiden
tt-mm-ss.txt

		let vFilTid_tmp3 = left ('$(vFilTid_tmp2)',8);  //henter ut tiden tt-
mm-ss

		let vFilTid = Replace('$(vFilTid_tmp3)','-',':'); //sluttformat
hh:mm:ss


		let vTidStempe1_tmp = '$(vFilDato)' & ' ' & '$(vFilTid)';

		let vFilTid = Timestamp('$(vFilTid_tmp2)','DD-MM-ÅÅÅÅ-hh-mm-ss');


		//Henter data fra filen

		test:

			LOAD @1 as SplDel

			FROM

			$(vFilNavn)

			(txt, utf8, no labels, delimiter is \x7f, msq, no eof);


		//Bruker tabell fra test1 til å splitte opp ord, omsepid, pasid oa.
Sletter tabellen test

		test2:

			load *,

//			replace(subfield(@1,' '),',','') as ord,

			peek('Name',$(i),'Files') as Name,

			Replace('$(vFilDato_tmp2)','-','.') as FilDato,

			Replace('$(vFilTid_tmp3)','-',':') as FilTid,
```

```
                    Timestamp#('$(vTidStempe1_tmp)','DD.MM.YYYY hh:mm:ss[.fff]') as
FilTidsstempel,

                    SubField('$(vFilNavn)','\',5) as FilNavn,

                    '$(FoldNo)' as prev_dato

                    Resident test;

                    drop Table test;


             //Ordene(tokens) som finnes      i tekstene lagres til store bokstaver.
Tabellen test2 slettes

             test3:

                    load *,

                    SubField(FilNavn,'_',1) as FilID,

                    SubField(FilNavn,'_',2) as PasID,

                    SubField(FilNavn,'_',3) as OmsEP,

                    SubField(FilNavn,'_',4) as DokType


                    Resident test2;

                    drop table test2;


NEXT i


//Lagrer tabellen test3 til en egen fil. Filen er interformat til QlikView. Sletter
deretter tabellen test3

store test3 into qvd\SPL-Dokumenter.qvd(qvd);

//drop table test3;

//drop table Files;

end sub

///$tab Call funksjoner

////Funksjon for å lese filer i en katalog


//For del 8.1.1.3 steg 1 i oppgaven

call DoDir('E:\Master_data\Data\2spldok\');

call LesFiler;


//For del 8.1.2.1 i oppgaven

call DoDir('E:\Annoterte-cvk-funn\');
```

```
call LesFiler;

//Kataloger for del 8.1.4.3
call DoDir('E:\Master_data\Data\6-Innhold_Del6\Dokumenter\Del6_data\');
call LesFiler;

//Katalog for del 8.1.4.3
call DoDir('E:\Master_data\Data\6-Innhold_Del6\Dokumenter\Del6_data\');
call LesFiler
```

Vedlegg 6: Flytte data

```
///$tab Main

SET ThousandSep=' ';

SET DecimalSep=',';

SET MoneyThousandSep=' ';

SET MoneyDecimalSep=',';

SET MoneyFormat='kr # ##0,00;kr -# ##0,00';

SET TimeFormat='hh:mm:ss';

SET DateFormat='DD.MM.YYYY';

SET TimestampFormat='DD.MM.YYYY hh:mm:ss[.fff]';

SET MonthNames='jan;feb;mar;apr;mai;jun;jul;aug;sep;okt;nov;des';

SET DayNames='ma;ti;on;to;fr;lø;sø';

//let vFraData = 'E:\Master_data\Data\0Raadata\';

//let vTilData = 'E:\Master_data\Data\3HentSplDok_m_mal\dokumenter\';

///$tab SplDok

//Flytter alle dokumenter som har mal fra spl-dokumentasjon fra katalog for alle
dokumenter til egen katalog

sub spldok(l_fra,l_til)

splData:

LOAD

 Distinct FilID

FROM

[E:\Master_data\Data\3HentSplDok_m_mal\qvd\SPL-Dokumenter.qvd]

(qvd);

let vRader = NoOfRows('splData')-1;

for c = 0 to vRader

       let vID = Peek('FilID',c,'splData');

       EXECUTE cmd.exe /c copy $(l_fra)$(vID)* $(l_til);

       next c;

end Sub

///$tab Call funksjoner

//Flytter data fra en katalog til en annen.

//Parametre for del 8.1.1.3 steg 2 i oppgaven.
```

```
call
spldok('E:\Master_data\Data\0Raadata\Dokumenter\','E:\Master_data\Data\3HentSplDok_m
_mal\dokumenter\');
```

Vedlegg 7: Les annoterte data

```
///$tab Main
SET ThousandSep=' ';
SET DecimalSep=',';
SET MoneyThousandSep=' ';
SET MoneyDecimalSep=',';
SET MoneyFormat='kr # ##0,00;kr -# ##0,00';
SET TimeFormat='hh:mm:ss';
SET DateFormat='DD.MM.YYYY';
SET TimestampFormat='DD.MM.YYYY hh:mm:ss[.fff]';
SET MonthNames='jan;feb;mar;apr;mai;jun;jul;aug;sep;okt;nov;des';
SET DayNames='ma;ti;on;to;fr;lø;sø';

//For del 8.1.2.2 i dokumenet
LOAD Tag,
     Beskrivelse,
     Tekst,
     Name,
     FilDato,
     FilTid,
     FilTidsstempel,
     OmsEP,
     PasID,
     desc,
     DokType,
     Info1,
     Info2,     1 as test,
     subfield(Name, '\' ,3) as FilNavn,
     subfield(Info1, '_' ,1) as Nr
FROM
E:\Master_data\Data\4Manuell_gjennomgang_careCVC\qvd\Annoterte_data.qvd(qvd);
```

Vedlegg 8: Les inn manuell gjennomgang av data

```
///$tab Main
SET ThousandSep=' ';
SET DecimalSep=',';
SET MoneyThousandSep=' ';
SET MoneyDecimalSep=',';
SET MoneyFormat='kr # ##0,00;kr -# ##0,00';
SET TimeFormat='hh:mm:ss';
SET DateFormat='DD.MM.YYYY';
SET TimestampFormat='DD.MM.YYYY hh:mm:ss[.fff]';
SET MonthNames='jan;feb;mar;apr;mai;jun;jul;aug;sep;okt;nov;des';
SET DayNames='ma;ti;on;to;fr;lø;sø';

//For del 8.1.2.3 i dokumentet
CareCVC:
LOAD FilNavn,
     Tag,
     desc,
     Tekst,
     FilNavn1,
     Tekst1,
     CVKInfo,
     IkkeAktuellt,
     Del1,
     Del2,
     Del3,
     [A:],
     [B:],
     Del4,
     Del5,
     Del6,
     Del7,
     Del8,
```

```
     Del9,
     Del10,
     Del11,
     Del12,
     Grunn,
     SubField(FilNavn,'_',1) as FilID
FROM
[E:\Master_data\Data\4Manuell_gjennomgang_careCVC\3CareCVC-gjennomgang_Original.xls]
(biff, embedded labels, table is Sheet1$);
//where IkkeAktuellt = 0;
//CVKInfo <> 0;

//store CareCVC into qvd/careCVC-spldok.qvd(qvd);
//drop table CareCVC;
```

Vedlegg 9: Hud/vev/sår

```
///$tab Main

SET ThousandSep=',';

SET DecimalSep='.';

SET MoneyThousandSep=',';

SET MoneyDecimalSep='.';

SET MoneyFormat='£#,##0.00;-£#,##0.00';

SET TimeFormat='hh:mm:ss';

SET DateFormat='DD/MM/YYYY';

SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';

SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';


//For del 8.1.3 dokumentet

let vFraData = 'E:\Master_data\Data\3HentSplDok_m_mal\dokumenter\';

let vTilData = 'E:\Master_data\Data\5-Del6-filer\SplDok_Med_Del6\';


Spl_del_tmp:

LOAD SplDel,

     Name,

     FilDato,

     FilTid,

     FilTidsstempel,

     OmsEP,

     PasID,

     DokType,

     FilNavn,

     prev_dato,

     FilID,

     FilID as splID,

     //Samler alle måter å skrive Hud/ves/sår på og lager en variabel for disse.
Disse kommer fra Lastdata_spldok_m_mal fra del 3

     if(SplDel = 'Hud/vev/sår',1,if(SplDel = 'Hud/vev/sår ',1,if(SplDel =
'Hud/vev/sår  ',1,if(SplDel = 'Hud/vev/sår   ',1,if(SplDel =
'Hud/vev/sår  ,',1,if(SplDel = 'Hud/vev/sår,',1,if(SplDel =
'Hud/vev/sår.',1,if(SplDel = 'Hud/vev/sår.Tørr',1,if(SplDel =
```

```
'Hud/vev/sår:',1,if(SplDel = 'Hud/vev/sår:Tørr',1,if(SplDel =
'Hud/vev/sår:Tørr/varm.  ',1,if(SplDel = 'Hud/vev/sår_',1,0))))))))))) as Del6

FROM

[E:\Master_data\Data\3HentSplDok_m_mal\qvd\SPL-Dokumenter.qvd]

(qvd);


Spl_del:

load *,1 as test

Resident Spl_del_tmp

where Del6 = 1;

drop Table Spl_del_tmp;


let vRader = NoOfRows('Spl_del')-1;

for c = 0 to vRader

      let vID = Peek('FilID',c,'Spl_del');


      EXECUTE cmd.exe /c copy $(vFraData)$(vID)* $(vTilData);

      next c;
```

Vedlegg 10: Keep text

```
# -*- coding: utf-8 -*-
import codecs
import time
import locale
import string
import os
import pickle
start_time = time.time()
#Variabler for data. Tilsvarer 8.1.4.2 i dokumenet
InnData = 'E:\Master_data\\Data\\6-Innhold_Del6\\Dokumenter\\Orddeling'
UtData = 'E:\Master_data\\Data\\6-Innhold_Del6\\Dokumenter\\Del6_data'



#henter filer i in katalog
def hent_filnavn(sti):
    liste = os.listdir(InnData)
    return liste

filliste = hent_filnavn(InnData)
AntFiler = len(filliste)
x = 0

while x < AntFiler:
    fil = InnData + "/" + filliste[x]
    buffer = []
    y=0
    keepCurrentSet = False
    #Henter data fra filene
    with open(fil) as fp:
        test = [i for i in fp]
        #test = fp.read().split()
    #test = test.strip()
    lengde = len(test)
```

```
    #print lengde, test


    while y < lengde:

           #Stopper skriving til buffer dersom vi går over i en ny del av spl
dokumentet

        if((test[y-1].startswith('7') and test[y].startswith('Aktivitet/')) or
(test[y].startswith('Aktivitet/funksjon')) or (test[y-1].startswith('8') and
test[y].startswith('Smerte/søvn/hvile/velvære')) or (test[y-1].startswith('9') and
test[y].startswith('Seksualitet/reproduksjon')) or (test[y-1].startswith('10') and
test[y].startswith('Sosialt/planlegging')) or (test[y-1].startswith('11') and
test[y].startswith('Åndelig/kulturelt/livsstil')) or (test[y-1].startswith('12') and
test[y].startswith('Annet/legedelegerte'))):

            keepCurrentSet = False

        #Hvis vi er inne i del 6 skrives ordene inn i bufferen

        if keepCurrentSet:

            temp = test[y]

            temp2 = temp.lstrip(' ')

            buffer.append(temp2)

            #buffer.append(test[y])

        #Sjekker om forrige ord var 6 og aktuelt ord er hud/vev/sår og setter flagg
til ta vare på data

        if ((test[y-1].startswith('6') and test[y].startswith('Hud/vev')) or
(test[y].startswith('Hud/vev/sår'))):

            keepCurrentSet = True

        #Sjekker at teller ikke går ut over index

        z = y+2

             #Stopper skriving til buffer hvis neste ord er et tall, 1-12, og
påfølgende ord er et nøkkelord i spl dokumenet

        if ( (z < lengde and test[y+1].startswith('7') and
test[y+2].startswith('Aktivitet/funksjon')) or (z < lengde and
test[y+1].startswith('8') and test[y+2].startswith('Smerte/søvn/hvile/velvære')) or
(z < lengde and test[y+1].startswith('9') and
test[y+2].startswith('Seksualitet/reproduksjon')) or (z < lengde and
test[y+1].startswith('10') and test[y+2].startswith('Sosialt/planlegging')) or (z <
lengde and test[y+1].startswith('11') and
test[y+2].startswith('Åndelig/kulturelt/livsstil')) or (z < lengde and
test[y+1].startswith('12') and test[y+2].startswith('Annet/sykepleiedelegerte')) or
(z < lengde and test[y+1].startswith('12') and
test[y+2].startswith('Annet/legedelegerte'))):

            keepCurrentSet = False

            #print test[y-1], test[y]

        y+=1

       #Skriver data til fil

    deltekst = ' '.join(buffer)
```

```
    clean = deltekst.replace('\n',' ')
    output_file = UtData + "/" + filliste[x]
    output = open(output_file,'w')
    output.write(clean)
    output.close()
    #clean = ''
    x+=1
print time.time() - start_time, "Sekunder"
```

Vedlegg 11: Flytte tomme filer

```
///$tab Main
SET ThousandSep=',';
SET DecimalSep='.';
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='£#,##0.00;-£#,##0.00';
SET TimeFormat='hh:mm:ss';
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

let vFraData = 'E:\Master_data\Data\6-Innhold_Del6\Dokumenter\Del6_data\';
let vTilData = 'E:\Master_data\Data\6-Innhold_Del6\Dokumenter\Ekskludert\';

//Leser inn filer som ikke har noe innhold i del 6.
data:
LOAD FilID
FROM
[E:\Master_data\Data\6-Innhold_Del6\Eksluderte_filer.xls]
(biff, embedded labels, table is Sheet1$);

//Flytter filene fra katalogen Del6_data til Eksludert
let vRader = NoOfRows('data')-1;
for c = 0 to vRader
	let vID = Peek('FilID',c,'data');

	EXECUTE cmd.exe /c move $(vFraData)$(vID)* $(vTilData);
	//i = i+1;
	next c;
```

Vedlegg 12: Dokumenter med mal

```
///$tab Main
SET ThousandSep=' ';
SET DecimalSep=',';
SET MoneyThousandSep=' ';
SET MoneyDecimalSep=',';
SET MoneyFormat='kr # ##0,00;kr -# ##0,00';
SET TimeFormat='hh:mm:ss';
SET DateFormat='DD.MM.YYYY';
SET TimestampFormat='DD.MM.YYYY hh:mm:ss[.fff]';
SET MonthNames='jan;feb;mar;apr;mai;jun;jul;aug;sep;okt;nov;des';
SET DayNames='ma;ti;on;to;fr;lø;sø';

//Tilhører 8.1.5.1 i dokumentet

//Viser innhold i dokumenter som har data i del 6 (hud/vev/sår).
//let vSet11 = 'E:\Master_data\Data\6-Innhold_Del6\Dokumenter\Del6_data\';

//Henter filer i Korpa som skal hente rådatafiler senere
let vSet11 = 'E:\Master_data\Data\6-Innhold_Del6\Dokumenter\Del6_data\';

//Kopierer filer som skal strippes for mal headere og data før male
let vFraData = 'E:\Master_data\Data\6-Innhold_Del6\Dokumenter\Orddeling\';
let vTilData = 'E:\Master_data\Data\7-AltInnhold_Mal\Raafiler\';
///$tab Oversikt filer
//Lager en liste over filene som har mal fra spl dokumenter i seg
sub DoDir(Root)
	let FoldNo = mid(Root,12, 10 ); //returns 'cd'. //SubField(Root,'\',2);
	FOR Each Ext in 'txt' //filtype å lete etter
		FOR Each File in FileList(Root & '*.' & Ext)
			Files:
			load '$(File)' as Name,
			FileTime('$(File)') as FileTime,
			RangeSum(Peek('FileCount'),1) as FileCount
```

```
                AutoGenerate 1;
            NEXT File
      NEXT Ext
      For Each Dir in DirList (Root & '*')    //leter i underkataloger
            Call DoDir(Dir)
      NEXT Dir
end sub
///$tab Les inn filer
//Leser inn data for alle filer i hver katalog
sub LesFiler
      let vRader = NoOfRows('Files')-1; //Finner antall filer som er lest
      for i=0 to vRader
            let vFilNavn = peek('Name',i,'Files'); //Henter filnavn


            //Henter Tidsstempel fra filnavn
            let vFilTid_tmp = right('$(vFilNavn)',23); //henter ut siste del av
filen. tidsstempel + .txt (dd-mm-åååå-tt-mm-ss.txt)
            let vFilDato_tmp2 = left('$(vFilTid_tmp)',10);  //henter ut dd-mm-åää
            let vFilDato = Replace('$(vFilDato_tmp2)','-','.'); //dato på format
dd.mm.åååå


            let vFilTid_tmp2 = right('$(vFilNavn)',12);          //henter ut tiden
tt-mm-ss.txt
            let vFilTid_tmp3 = left ('$(vFilTid_tmp2)',8);  //henter ut tiden tt-
mm-ss
            let vFilTid = Replace('$(vFilTid_tmp3)','-',':'); //sluttformat
hh:mm:ss


            let vTidStempe1_tmp = '$(vFilDato)' & ' ' & '$(vFilTid)';
            let vFilTid = Timestamp('$(vFilTid_tmp2)','DD-MM-ÅÅÅÅ-hh-mm-ss');


            //Henter data fra filen
            test:
                  LOAD @1 as SplDel
                  FROM
                  $(vFilNavn)
                  (txt, utf8, no labels, delimiter is \x7f, msq, no eof);
```

```
            //Bruker tabell fra test1 til å splitte opp ord, omsepid, pasid oa.
Sletter tabellen test

            test2:

                  load *,

                  peek('Name',$(i),'Files') as Name,

                  Replace('$(vFilDato_tmp2)','-','.') as FilDato,

                  Replace('$(vFilTid_tmp3)','-',':') as FilTid,

                  Timestamp#('$(vTidStempe1_tmp)','DD.MM.YYYY hh:mm:ss[.fff]') as
FilTidsstempel,

                  SubField('$(vFilNavn)','\',7) as FilNavn,

                  '$(FoldNo)' as prev_dato

                  Resident test;

                  drop Table test;


            //Ordene(tokens) som finnes      i tekstene lagres til store bokstaver.
Tabellen test2 slettes

            test3:

                  load *,

                  SubField(FilNavn,'_',1) as FilID,

                  SubField(FilNavn,'_',2) as PasID,

                  SubField(FilNavn,'_',3) as OmsEP,

                  SubField(FilNavn,'_',4) as DokType


                  Resident test2;

                  drop table test2;


NEXT i


//Lagrer tabellen test3 til en egen fil. Filen er interformat til QlikView. Sletter
deretter tabellen test3

store test3 into qvd\SPL-Dokumenter.qvd(qvd);

//drop table test3;

//drop table Files;

end sub

///$tab LesInn og kopier data

sub les
```

```
data:
LOAD
     FilID
FROM
[E:\Master_data\Data\7-AltInnhold_Mal\qvd\SPL-Dokumenter.qvd](qvd);

//Flytter filene fra katalogen Del6_data til Eksludert
let vRader = NoOfRows('data')-1;
for c = 0 to vRader
       let vID = Peek('FilID',c,'data');

       EXECUTE cmd.exe /c copy $(vFraData)$(vID)* $(vTilData);
       //i = i+1;
       next c;



end Sub
///$tab Call funksjoner
//////Data for å hente ut info om alle spl dokumenter
//call DoDir(vSet11);
//call LesFiler;

call les
```

Vedlegg 13: Fjern mal overskrifter

```
# -*- coding: utf-8 -*-
import codecs
import time
import locale
import string
import os
import pickle
start_time = time.time()

#Variabler for data
InnData = 'E:\Master_data\\Data\\7-AltInnhold_Mal\\Raafiler'
UtData = 'E:\Master_data\\Data\\7-AltInnhold_Mal\\Alt_Innhold_i_Mal'

#henter filer i in katalog
def hent_filnavn(sti):
    liste = os.listdir(InnData)
    return liste

filliste = hent_filnavn(InnData)
AntFiler = len(filliste)
x = 0

while x < AntFiler:
    fil = InnData + "/" + filliste[x]
    buffer = []
    y=0
    keepCurrentSet = False
    #Henter data fra filene
    with open(fil) as fp:
        test = [i for i in fp]
    lengde = len(test)

    while y < lengde:
```

```
        #Hvis vi er inne i en av delene til malen skrives ordene inn i bufferen

        if keepCurrentSet:

            temp = test[y]

            #temp2 = temp.strip(' ')

            buffer.append(temp)

            #buffer.append(test[y])

        if((test[y-1].startswith('1') and test[y].startswith('Kommunikasjon/sans'))
or (test[y-1].startswith('2') and test[y].startswith('Kunnskap/utvikling/psykisk'))
or

           (test[y-1].startswith('3') and
test[y].startswith('Åndedrett/Sirkulasjon')) or (test[y].startswith('A:Åndedrett'))
or (test[y].startswith('B:Sirkulasjon')) or

           (test[y-1].startswith('4') and test[y].startswith('Ernæring/væske')) or
(test[y-1].startswith('5') and test[y].startswith('Eliminasjon')) or (test[y-
1].startswith('6') and test[y].startswith('Hud/vev/')) or

           (test[y-1].startswith('7') and test[y].startswith('Aktivitet/funksjon'))
or (test[y-1].startswith('8') and test[y].startswith('Smerte/søvn/')) or (test[y-
1].startswith('10') and test[y].startswith('Sosialt/planlegging')) or

           (test[y-2].startswith('12') and test[y-
1].startswith('Annet/legedelegerte') and test[y].startswith('oppgaver')) or (test[y-
1].startswith('9') and test[y].startswith('Seksualitet/reproduksjon')) or

           (test[y-1].startswith('11') and
test[y].startswith('Åndelig/kulturelt/'))):#(test[y-1].startswith('7') and
test[y].startswith('Aktivitet/funksjons-status')) or (test[y-1].startswith('8') and
test[y].startswith('Smerte/søvn/hvile/velvære')) or (test[y-1].startswith('9') and
test[y].startswith('Seksualitet/reproduksjon')) or (test[y-1].startswith('10') and
test[y].startswith('Sosialt/planlegging')) or (test[y-1].startswith('11') and
test[y].startswith('Åndelig/kulturelt/livsstil')) or (test[y-1].startswith('12') and
test[y].startswith('Annet/legedelegerte'))):

            keepCurrentSet = True

        #Sjekker at teller ikke går ut over index

        z = y+2

             #Stopper skriving til buffer hvis neste ord er et tall, 1-12, og
påfølgende ord er et nøkkelord i spl dokumenet

        if ( (z < lengde and test[y+1].startswith('2') and
test[y+2].startswith('Kunnskap/utvikling/psykisk')) or (z < lengde and
test[y+1].startswith('3') and test[y+2].startswith('Åndedrett/Sirkulasjon'))

             or (z < lengde and test[y+1].startswith('A:Åndedrett')) or (z < lengde
and test[y+1].startswith('B:Sirkulasjon')) or (z < lengde and
test[y+1].startswith('4') and test[y+2].startswith('Ernæring/væske'))

             or (z < lengde and test[y+1].startswith('5') and
test[y+2].startswith('Eliminasjon')) or (z < lengde and test[y+1].startswith('6')
and test[y+2].startswith('Hud/vev/'))

             or (z < lengde and test[y+1].startswith('7') and
test[y+2].startswith('Aktivitet/funksjon')) or (z < lengde and
test[y+1].startswith('8') and test[y+2].startswith('Smerte/søvn/'))

             or (z < lengde and test[y+1].startswith('10') and
test[y+2].startswith('Sosialt/planlegging')) or (z < lengde and
```

```
test[y+1].startswith('12') and test[y+2].startswith('Annet/legedelegerte') and
test[y+3].startswith('oppgaver'))

              or (z < lengde and test[y+1].startswith('9') and
test[y+2].startswith('Seksualitet/reproduksjon')) or (z < lengde and
test[y+1].startswith('11') and test[y+2].startswith('Åndelig/kulturelt/'))):#(z <
lengde and test[y+1].startswith('7') and test[y+2].startswith('Aktivitet/funksjons-
status')) or (z < lengde and test[y+1].startswith('8') and
test[y+2].startswith('Smerte/søvn/hvile/velvære')) or (z < lengde and
test[y+1].startswith('9') and test[y+2].startswith('Seksualitet/reproduksjon')) or
(z < lengde and test[y+1].startswith('10') and
test[y+2].startswith('Sosialt/planlegging')) or (z < lengde and
test[y+1].startswith('11') and test[y+2].startswith('Åndelig/kulturelt/livsstil'))
or (z < lengde and test[y+1].startswith('12') and
test[y+2].startswith('Annet/legedelegerte'))):

            keepCurrentSet = False

        y+=1

       #Skriver data til fil

    deltekst = ' '.join(buffer)

    clean = deltekst.replace('\n',' ')

    output_file = UtData + "/" + filliste[x]

    output = open(output_file,'w')

    output.write(clean)

    output.close()

    #clean = ''

    x+=1

print time.time() - start_time, "Sekunder"
```

Vedlegg 14: Splitte dokumenter (CareCVC og UtenCareCVC)

```
///$tab Main

SET ThousandSep=',';

SET DecimalSep='.';

SET MoneyThousandSep=',';

SET MoneyDecimalSep='.';

SET MoneyFormat='£#,##0.00;-£#,##0.00';

SET TimeFormat='hh:mm:ss';

SET DateFormat='DD/MM/YYYY';

SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';

SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

//Viser til punkt 8.1.6 i dokumentet

////Kun Del6 data

//let vFraData = 'E:\Master_data\Data\6-Innhold_Del6\Dokumenter\Del6_data\';

//let vTilData = 'E:\Master_data\Data\8-
CareCVC_UtenCareCVC\Dokumenter\Kun_Del6\CareCVC\';

//Alt innhold i filer

let vFraData = 'E:\Master_data\Data\7-AltInnhold_Mal\Alt_Innhold_i_Mal\';

let vTilData = 'E:\Master_data\Data\8-
CareCVC_UtenCareCVC\Dokumenter\Alt_innhold\CareCVC\';

CareCVC:

LOAD

     FilID

FROM

[E:\Master_data\Data\4Manuell_gjennomgang_careCVC\qvd\careCVC-spldok.qvd](qvd)

where CVKInfo = 6;


let vRader = NoOfRows('CareCVC')-1;

for c = 0 to vRader

       let vID = Peek('FilID',c,'CareCVC');


       EXECUTE cmd.exe /c move $(vFraData)$(vID)* $(vTilData);

       //i = i+1;

       next c;
```

Vedlegg 15: Frekvensordliste

```
# -*- coding: utf-8 -*-

#Kopier filen til C:\Python27\Lib

#starter filen med å skrive from Setninger import*

import time

start_time = time.time()

import nltk.data

import string

import os

from nltk.tokenize import word_tokenize


tokenizer = nltk.data.load('tokenizers/punkt/norwegian.pickle')


#Variabler for hvor data befinner seg. Skal lage ordliste for 4 datasett.

#data_sti = 'E:\Master_data\\Preprosessering\\0-Raadata\\Kun_Del6\\CareCVC'

#data_sti = 'E:\Master_data\\Preprosessering\\0-Raadata\\Kun_Del6\\UtenCareCVC'

#data_sti = 'E:\Master_data\\Preprosessering\\0-Raadata\\Alt_innhold\\UtenCareCVC'

#data_sti = 'E:\Master_data\\Preprosessering\\0-Raadata\\Alt_innhold\\CareCVC'


#Variabler for frekvensordlister etter forkortelser

#data_sti = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Kun_Del6\\CareCVC'

#data_sti = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Kun_Del6\\UtenCareCVC'

#data_sti = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Alt_innhold\\CareCVC'

#data_sti = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Alt_innhold\\UtenCareCVC'


#Variabler for frekvensordlister etter stemming

#data_sti = 'E:\Master_data\\Preprosessering\\3-
Stemming\Dokumenter\\stemming\\Kun_Del6\\CareCVC'

#data_sti = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\stemming\\Kun_Del6\\UtenCareCVC'

#data_sti = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\stemming\\Alt_innhold\\CareCVC'

data_sti = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\stemming\\Alt_innhold\\UtenCareCVC'
```

```
FrekvensSti = 'E:\Master_data\\Preprosessering\\1-Frekvensordliste_raadata'

#henter filer i in katalog
def hent_filnavn(sti):
    liste = os.listdir(data_sti)
    return liste

filliste = hent_filnavn(data_sti)
AntFiler = len(filliste)
x = 0
AlleOrd = []

#Variabel for alle setninger som er lest inn
alle_setninger = ''

while x < AntFiler:
    fil = data_sti + "\\" + filliste[x]
    with open(fil) as fp:
        para = fp.read()
    setning = tokenizer.tokenize(para)
    words = para.split()
    AlleOrd = AlleOrd + words
    x+=1
l = [item.lower() for item in AlleOrd]
#m = [item.replace('"','') for item in l]
#n = [item.replace('(','') for item in m]

#print l
from nltk.probability import FreqDist
fd_sti = 'c:\data\AntOrdCorpa'

#Lager frekvensliste over ordene i alle_ord og skriver denne ut til fd filen
fd = FreqDist(l)
```

```
#Frekvensordlisten skrives ut til følgende plass
#output_file = FrekvensSti + "\\" + "Original_Del6_CareCVC.txt"
#output_file = FrekvensSti + "\\" + "Original_Del6_UtenCareCVC.txt"
#output_file = FrekvensSti + "\\" + "Original_Alt_UtenCareCVC.txt"
#output_file = FrekvensSti + "\\" + "Original_Alt_CareCVC.txt"

#Frekvensordlister etter forkortelser
#output_file = FrekvensSti + "\\" + "Forkortelser_Del6_CareCVC.txt"
#output_file = FrekvensSti + "\\" + "Forkortelser_Del6_UtenCareCVC.txt"
#output_file = FrekvensSti + "\\" + "Forkortelser_Alt_CareCVC.txt"
#output_file = FrekvensSti + "\\" + "Forkortelser_Alt_UtenCareCVC.txt"

#Frekvensordlister etter forkortelser
#output_file = FrekvensSti + "\\" + "Stemming_Del6_CareCVC.txt"
#output_file = FrekvensSti + "\\" + "Stemming_Del6_UtenCareCVC.txt"
#output_file = FrekvensSti + "\\" + "Stemming_Alt_CareCVC.txt"
output_file = FrekvensSti + "\\" + "Stemming_Alt_UtenCareCVC.txt"


output = open(output_file,'w')
for word in sorted(fd):
    pair = word + '    ->' + str(fd[word])
    output.write(pair+'\n')
output.close()

print time.time() - start_time, "Sekunder"
```

Vedlegg 16: Skrive ut forkortelser

```
# -*- coding: utf-8 -*-

import nltk

import string

import os

import re

import codecs

import time

start_time = time.time()


#Sti til data med Del6_CareCVC

DataSti = 'E:\Master_data\\Preprosessering\\0-Raadata\\Kun_Del6\\CareCVC'

OrdSti = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Kun_Del6\\CareCVC'


#Sti til data med Del6_UtenCareCVC

#DataSti = 'E:\Master_data\\Preprosessering\\0-Raadata\\Kun_Del6\\UtenCareCVC'

#OrdSti = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Kun_Del6\\UtenCareCVC'


#Sti til data med Alt_CareCVC

#DataSti = 'E:\Master_data\\Preprosessering\\0-Raadata\\Alt_innhold\\CareCVC'

#OrdSti = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Alt_innhold\\CareCVC'


#Sti til data med Alt_UtenCareCVC

#DataSti = 'E:\Master_data\\Preprosessering\\0-Raadata\\Alt_innhold\\UtenCareCVC'

#OrdSti = 'E:\Master_data\\Preprosessering\\2-
Forkortelser\\Dokumenter\\Alt_innhold\\UtenCareCVC'

pattern = r'''(?x)

    ([0-3][0-9][/.][0-1][0-9][/.][0-9][0-9]) #sjekker dato

   | ([0-2][0-9][/.][0-5][0-9]) #sjekk for tidspunkt

   | ([0-2][0-9][0-9][/][0-9][0-9]) #Blodtrykk

   | ([0-4][0-9][/.][0-9]) # sjekk for temeratur

   | ([0-9][/.][0-9]) # sjekk for dosering

   | ([A-Z][0-9][0-9][/.][0-9]) #Sjekk for diagnoser
```

```
   | ([A-Z]\.)+ #forkortelser, eksempel er U.S.A

   | \w+(-\w+)* # Ord med mulige apostrofer

   | \W+(-\W+)* # Ord med mulige apostrofer

   | [][.,;"'?():-_`]

    '''


def hent_filnavn(sti):

    liste = os.listdir(DataSti)

    return liste


filliste = hent_filnavn(DataSti)

AntFiler = len(filliste)

x = 0

AlleOrd = []

SmaaOrd = []

while x < AntFiler:

    fil = DataSti + "\\" + filliste[x]

    #with open(LoggSti,"w") as logg:

        #logg.write(filliste[x] + '\n')

    with codecs.open(fil, 'rb', 'UTF-8') as fp:

        setning = fp.readlines()

    ut_fil = OrdSti + "\\" + filliste[x]

    AntSetninger = len(setning)

    with open(ut_fil,'w') as output_file:

        i=0

        while i < AntSetninger:

            ut = setning[i]# + '\n'  kommentert ut 09.02

            sentence = re.sub('CVK-|-CVK|C\.VK|cvk-|Cvk-|Cvk|Cvk|CvK|CVk-
|CVk|CVKn|CVKen|CVK_|cvk|CVK`en| cvk:n | CVK:n|CVK-| CVK\'n | CVK`n|
cvk:n|CVK\'|CVK!|CVK\,| CVK\'en|CVK; |CVK:| CVK?| CVK/| CVK\)',' CVK ',ut) #Fjerner
ord hvor CVK-blodig blir CVK blodig

            sentence2 = re.sub('VAP-|VAPen|VAPn|VApen| Vap | vap | vap\. |vap-
|vapn|VAP\'en| Vap\'en | VAP:|VAP;',' VAP ',sentence) #Skriver VAP på en uniform
måte

            sentence3 = re.sub('PICC-|picc-|Picc-| picc |PICC_',' PICC ',sentence2)
#Erstatter PICC-line med PICC line

            sentence4 = re.sub('piccline|PICCLINE|PICCline|Piccline|Pickline','PICC
line ',sentence3) #Fjerner ord hvor CVK-blodig blir CVK blodig
```

```
            sentence5 =
re.sub('Hickamann|hickmanns|Hickmann|hickmann|Hickmans|Hockamm|hickman|Hicmann|hicma
nn|Hickman|Hikmans|Hicman|Hikman|HIcmann',' Hickman ',sentence4) #Fjerner ord hvor
CVK-blodig blir CVK blodig

            sentence6 = re.sub(' \.kat ',' kateter ',sentence5) #Fjerner ord hvor
CVK-blodig blir CVK blodig

            sentence7 = re.sub('"|\(|\)|/|!|%|\+|\'',' ',sentence6)


            sentence8 = re.sub(' v/CVK',' ved CVK',sentence7)

            sentence9 = re.sub('CVKstell','CVK stell ',sentence8) #Fjerner ord hvor
CVK-blodig blir CVK blodig

            sentence10 = re.sub('-|:|;',' ',sentence9)

            sentence11 = re.sub('Dr ','Doktor ',sentence10)

            sentence12 = re.sub(' Dr\. | dr. ',' doktor ',sentence11)


            sentence13 = re.sub(u' hø\. | hø\.| Hø\. ',u' høyre ',sentence12)


            sentence14 = re.sub(' opr | opr\.| Opr | Opr\.| op | op\.',' operasjon
',sentence13)

            sentence15 = re.sub('Opr |Op |Op\.','Operasjon ',sentence14)


            sentence16 = re.sub('Pas |Pas\. |PAs |pas |Pas ','Pasient ',sentence15)


            sentence17 = re.sub(' Pas\.| pas\.| pas ',' pasient ',sentence16)


            sentence18 = re.sub(' PAS\.',' pasient.',sentence17)


            sentence19 = re.sub('Sep\.|Sep ','Seponert ',sentence18)


            sentence20 = re.sub(' sep\.| sep ',' seponert ',sentence19)


            sentence21 = re.sub(' ua\.| u\.a\. ',' uten anmerkning ',sentence20)


            sentence22 = re.sub('U\.a\.|u\.a\.|u\.a','Uten anmerkning. ',sentence21)


            sentence23 = re.sub('U\.a\.|u\.a\.|u\.a','Uten anmerkning. ',sentence22)


            sentence24 = re.sub('U\.t|u\.t ','Undertegnede ',sentence23)


            sentence25 = re.sub(' u\.t\.| u\.t ',' undertegnede ',sentence24)


            sentence26 = re.sub('PVK`ene|\?PVK|PVKer',' PVK ',sentence25)


            sentence27 = re.sub(u'\.Smøring',u'. Smøring ',sentence26)
```

```
            sentence28 = re.sub('\.Trenger','. Trenger ',sentence27)


            sentence29 = re.sub(u'pasient\.Må',u'pasient. Må ',sentence28)

            sentence30 = re.sub('\.\.','\. ',sentence29)
            sentence31 = re.sub('dialysespl','dialyse sykepleier ',sentence30)

            sentence32 = re.sub(' Spl\.| spl | spl\.',' sykepleier ',sentence31)

            sentence33 = re.sub('i\.Har','i. Har ',sentence32)

            sentence34 = re.sub('i\.Har','i. Har ',sentence33)

            sentence35 = re.sub(u'\.ønsker',u'. Ønsker ',sentence34)

            sentence36 = re.sub(' ve | ve\.',' venstre ',sentence35)

            sentence37 = re.sub('skift\.da',' skift. Da ',sentence36)

            sentence38 = re.sub(u'sårhulene\.Fukte',u' sårhulene. Fukte
',sentence37)
            sentence39 = re.sub('rygg\.Har',' rygg. Har ',sentence38)

            sentence40 = re.sub(u'kavilon\.såret',u' kavilon. Såret ',sentence39)


            sentence41 = re.sub('fuktighet\.Det',' fuktighet. Det ',sentence40)


            sentence42 = re.sub('dragen\.Sivar',' dragen. Sivar ',sentence41)
            sentence43 = re.sub('baken\.Penslet',' baken. Penslet ',sentence42)

            sentence44 = re.sub('albue\.Penslet',' albue. Penslet ',sentence43)

            sentence45 = re.sub(u'pasient\.Må',u' pasient. Må ',sentence44)

            sentence46 = re.sub('\*\w',' \w',sentence45)
            sentence47 = re.sub('\.\w','. \w',sentence46)

            sentence48 = re.sub('\.','',sentence47)
            sentence49 = re.sub('\?','',sentence48)
            sentence50 = re.sub('\\\\','',sentence49)

            words = nltk.regexp_tokenize(sentence50,pattern)
```

```
            sentence1000 = ''.join(words)
            ut2 = sentence1000 #+ "\n"
            output_file.write(ut2.encode("UTF-8"))
            i+=1
    x += 1
print time.time() - start_time, "Sekunder"
```

Vedlegg 17: Omgjøring til små bokstaver

```
# -*- coding: utf-8 -*-

from nltk.stem import SnowballStemmer

import codecs

import time

import locale

import string

import os

import pickle

start_time = time.time()


#For Del6_CareCVC data

#InnData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\orddeling\\Kun_Del6\\CareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\Smaaord\\Kun_Del6\\CareCVC'


#For Del6_UtenCareCVC

#InnData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\orddeling\\Kun_Del6\\UtenCareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\Smaaord\\Kun_Del6\\UtenCareCVC'


#For Alt_CareCVC data

#InnData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\orddeling\\Alt_innhold\\CareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\Smaaord\\Alt_innhold\\CareCVC'


#For Alt_UtenCareCVC

InnData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\orddeling\\Alt_innhold\\UtenCareCVC'

UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\Smaaord\\Alt_innhold\\UtenCareCVC'


norsk_stemmer = SnowballStemmer('norwegian')
```

```
#henter filer i in katalog
def hent_filnavn(sti):
    liste = os.listdir(InnData)
    return liste

filliste = hent_filnavn(InnData)
AntFiler = len(filliste)
x = 0

while x < AntFiler:
    fil = InnData + "/" + filliste[x]
    buffer = []
    y=0
    keepCurrentSet = False
    #Henter data fra filene
    with open(fil) as fp:
        test = [i for i in fp]
    lengde = len(test)

    while y < lengde:
        ord = test[y]
        ord2 = ord.lower()
        buffer.append(ord2)
        y+=1
       #Skriver data til fil
    deltekst = ' '.join(buffer)
    clean = deltekst.replace('\n',' ')
    output_file = UtData + "/" + filliste[x]
    output = open(output_file,'w')
    output.write(clean)
    output.close()
    #clean = ''
    x+=1
print time.time() - start_time, "Sekunder"
```

Vedlegg 18: Stemming

```
# -*- encoding:utf-8 -*-

from nltk.stem import SnowballStemmer

import nltk.data

from nltk.tokenize import word_tokenize

tokenizer = nltk.data.load('tokenizers/punkt/norwegian.pickle')

import re

import codecs

import time

import locale

import string

import os

import pickle


start_time = time.time()


#For data Del6_CareCVC

#InnData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\Smaaord\\Kun_Del6\\CareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\stemming\\Kun_Del6\\CareCVC'


#For data Del6_UtenCareCVC

#InnData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\Smaaord\\Kun_Del6\\UtenCareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\stemming\\Kun_Del6\\UtenCareCVC'


#For data Alt_CareCVC

#InnData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\Smaaord\\Alt_innhold\\CareCVC'

#UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\stemming\\Alt_innhold\\CareCVC'


#For data Alt_UtenCareCVC

InnData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\Smaaord\\Alt_innhold\\UtenCareCVC'
```

```
UtData = 'E:\Master_data\\Preprosessering\\3-
Stemming\\Dokumenter\\stemming\\Alt_innhold\\UtenCareCVC'


def hent_filnavn(sti):

    liste = os.listdir(InnData)

    return liste

filliste = hent_filnavn(InnData)

AntFiler = len(filliste)

x = 0



norsk_stemmer = SnowballStemmer('norwegian')



while x < AntFiler:

    words3 = []

    fil = InnData + "/" + filliste[x]

    y=0

    #Henter data fra filene

    with open(fil) as fp:

        para = [line.rstrip('\n') for line in fp]

        setning = para

        setninger = len(setning)

        i = 0

        while i < setninger:

            sentence = setning[i]

            sentence2 = re.sub(u'[%s]' % re.escape(string.punctuation), ' ',
sentence)

            words = word_tokenize(sentence)

            words2 = [w.lower() for w in words]

            z=len(words)

            q=0

            while q<z:

                test = words[q]

                test3 = unicode(test,'utf-8')
```

```
                test2 = norsk_stemmer.stem(test3)
                words3.append(test2)
                q+=1
            i+=1
        output_file = UtData + "/" + filliste[x]
        deltekst = ' '.join(words3)
        with open(output_file,'wb') as f:
            f.write(deltekst.encode('utf-8'))
    x+=1
print time.time() - start_time, "Sekunder"
```

Vedlegg 19: Trenings- og testdata

```
///$tab Main

SET ThousandSep=' ';

SET DecimalSep=',';

SET MoneyThousandSep=' ';

SET MoneyDecimalSep=',';

SET MoneyFormat='kr # ##0,00;kr -# ##0,00';

SET TimeFormat='hh:mm:ss';

SET DateFormat='DD.MM.YYYY';

SET TimestampFormat='DD.MM.YYYY hh:mm:ss[.fff]';

SET MonthNames='jan;feb;mar;apr;mai;jun;jul;aug;sep;okt;nov;des';

SET DayNames='ma;ti;on;to;fr;lø;sø';


//Skal fordele data i 3 deler: trening, dev og test, med fordeling 60%, 20% og 20%

//Leser inn filer vi har i korpa og fordeler disse i test, trening og dev. Filnavn
er lik, men innholdet er ulikt.


//Henter inn filer som skal fordeles fra dokumenter med carecvc og uten fra
variablene under

let vSet11 = 'E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Alt_innhold\CareCVC\';

let vSet22 = 'E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Alt_innhold\UtenCareCVC\';


//Fordelingsfil for CareCVC

let vFordelingsfil_CareCVC = 'E:\Master_data\Treningsdata\Fordeling_CareCVC.xls';


//Fordelingsfil for UtenCareCVC

let vFordelingsfil_UtenCareCVC =
'E:\Master_data\Treningsdata\Fordeling_UtenCareCVC.xls';

///$tab Oversikt filer

//Lager en liste over filene som har mal fra spl dokumenter i seg

sub DoDir(Root)

	let FoldNo = mid(Root,12, 10 ); //returns 'cd'. //SubField(Root,'\',2);

	FOR Each Ext in 'txt' //filtype å lete etter

		FOR Each File in FileList(Root & '*.' & Ext)
```

```
                    Files:
                    load '$(File)' as Name,
                    FileTime('$(File)') as FileTime,
                    RangeSum(Peek('FileCount'),1) as FileCount
                    AutoGenerate 1;
             NEXT File
      NEXT Ext
      For Each Dir in DirList (Root & '*')    //leter i underkataloger
             Call DoDir(Dir)
      NEXT Dir
end sub
///$tab Les inn filer
//Leser inn data for alle filer i hver katalog
sub LesFiler
      let vRader = NoOfRows('Files')-1; //Finner antall filer som er lest
      for i=0 to vRader
             let vFilNavn = peek('Name',i,'Files'); //Henter filnavn

             //Henter Tidsstempel fra filnavn
             let vFilTid_tmp = right('$(vFilNavn)',23); //henter ut siste del av
filen. tidsstempel + .txt (dd-mm-åååå-tt-mm-ss.txt)
             let vFilDato_tmp2 = left('$(vFilTid_tmp)',10);  //henter ut dd-mm-ååå
             let vFilDato = Replace('$(vFilDato_tmp2)','-','.'); //dato på format
dd.mm.åååå

             let vFilTid_tmp2 = right('$(vFilNavn)',12);          //henter ut tiden
tt-mm-ss.txt
             let vFilTid_tmp3 = left ('$(vFilTid_tmp2)',8);  //henter ut tiden tt-
mm-ss
             let vFilTid = Replace('$(vFilTid_tmp3)','-',':'); //sluttformat
hh:mm:ss

             let vTidStempe1_tmp = '$(vFilDato)' & ' ' & '$(vFilTid)';
             let vFilTid = Timestamp('$(vFilTid_tmp2)','DD-MM-ÅÅÅÅ-hh-mm-ss');

             //Henter data fra filen
             test:
```

```
                    LOAD @1 as SplDel

                    FROM

                    $(vFilNavn)

                    (txt, utf8, no labels, delimiter is \x7f, msq, no eof);


             //Bruker tabell fra test1 til å splitte opp ord, omsepid, pasid oa.
Sletter tabellen test

             test2:

                    load *,

//                  replace(subfield(@1,' '),',','') as ord,

                    peek('Name',$(i),'Files') as Name,

                    Replace('$(vFilDato_tmp2)','-','.') as FilDato,

                    Replace('$(vFilTid_tmp3)','-',':') as FilTid,

                    Timestamp#('$(vTidStempe1_tmp)','DD.MM.YYYY hh:mm:ss[.fff]') as
FilTidsstempel,

                    SubField('$(vFilNavn)','_',3) as OmsEP, //Henter omsepid fra
filnavn

                    SubField('$(vFilNavn)','_',2) as PasID, //Henter pasid fra
filnavn

                    SubField('$(vFilNavn)','_',4) as DokType,     //Henter pasid fra
filnavn,

                    SubField('$(vFilNavn)','\',8) as FilNavn,

                    '$(FoldNo)' as prev_dato

                    Resident test;

                    drop Table test;


             //Ordene(tokens) som finnes      i tekstene lagres til store bokstaver.
Tabellen test2 slettes

             test3:

                    load *,

                    SubField(FilNavn,'_',1) as FilID

             //     Upper(ord) as OrdStore

                    Resident test2;

                    drop table test2;


NEXT i
```

```
//Lagrer tabellen test3 til en egen fil. Filen er interformat til QlikView. Sletter
deretter tabellen test3

//store test3 into qvd\SPL-Dokumenter.qvd(qvd);

//drop table test3;

drop table Files;

end sub

///$tab Fordeling

sub Fordeling (l_fra, l_til, l_fil, l_type)

Fordeling:

LOAD FilID,

     Fordeling

FROM

$(l_fil)

(biff, embedded labels, table is Sheet1$)

where Fordeling = '$(l_type)';//'Test';


let vRader = NoOfRows('Fordeling')-1;

for c = 0 to vRader

       let vID = Peek('FilID',c,'Fordeling');


       EXECUTE cmd.exe /c move $(l_fra)$(vID)* $(l_til);

       next c;

       drop table Fordeling;


end sub

///$tab Call funksjoner

//Løpenummer eksporteres ut til excel. I excel lages det en random funksjon som gir
et tall mellom 0 og 1.

//Kopierer resultat fra denne fordelingen, limer inn kun verdiene. Sorterer på ID og
random nummer i synkende rekkefølge.

//Fordelingen blir da

//Denne fordelingen blir gjentatt til man kommer til slutt i ID nr.

//Denne metoden gjelder for både dokumenter med careCVC tag og de uten.


////Data for å hente ut info om alle dokumenter annotert med carecvc.

//call DoDir(vSet11);
```

```
//call LesFiler;


//*****************************************************************************
*****************************************************************************
***********************************************

//Kommandoer for å fordele raadata

//Fordele data for Alt, carecvc

//call Fordeling ('E:\Master_data\Preprosessering\0-
Raadata\Alt_innhold\CareCVC\','E:\Master_data\Treningsdata\Dokumenter\1-
Raadata\Alt\Test\1-CareCVC\', '$(vFordelingsfil_CareCVC)', 'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\0-
Raadata\Alt_innhold\CareCVC\','E:\Master_data\Treningsdata\Dokumenter\1-
Raadata\Alt\Training\1-CareCVC\', '$(vFordelingsfil_CareCVC)','Trening' );



//Fordele data for Alt, UtenCareCVC

//call Fordeling ('E:\Master_data\Preprosessering\0-
Raadata\Alt_innhold\UtenCareCVC\','E:\Master_data\Treningsdata\Dokumenter\1-
Raadata\Alt\Test\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)', 'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\0-
Raadata\Alt_innhold\UtenCareCVC\','E:\Master_data\Treningsdata\Dokumenter\1-
Raadata\Alt\Training\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)', 'Trening' );



//Fordele data for Del6, CareCVC

//call Fordeling ('E:\Master_data\Preprosessering\0-
Raadata\Kun_Del6\CareCVC\','E:\Master_data\Treningsdata\Dokumenter\1-
Raadata\Del6\Test\1-CareCVC\', '$(vFordelingsfil_CareCVC)', 'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\0-
Raadata\Kun_Del6\CareCVC\','E:\Master_data\Treningsdata\Dokumenter\1-
Raadata\Del6\Training\1-CareCVC\', '$(vFordelingsfil_CareCVC)','Trening' );



//Fordele data for Del6, UtenCareCVC

//call Fordeling ('E:\Master_data\Preprosessering\0-
Raadata\Kun_Del6\UtenCareCVC\','E:\Master_data\Treningsdata\Dokumenter\1-
Raadata\Del6\Test\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)', 'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\0-
Raadata\Kun_Del6\UtenCareCVC\','E:\Master_data\Treningsdata\Dokumenter\1-
Raadata\Del6\Training\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)', 'Trening' );




//*****************************************************************************
*****************************************************************************
***********************************************

//Kommandoer for å fordele forkortelser

//Fordele data for Alt, carecvc
```

```
//call Fordeling ('E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Alt_innhold\CareCVC\','E:\Master_data\Treningsdata\Dokumente
r\2-forkortelser\Alt\Test\1-CareCVC\', '$(vFordelingsfil_CareCVC)', 'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Alt_innhold\CareCVC\','E:\Master_data\Treningsdata\Dokumente
r\2-forkortelser\Alt\Training\1-CareCVC\', '$(vFordelingsfil_CareCVC)','Trening' );



//Fordele data for Alt, UtenCareCVC

//call Fordeling ('E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Alt_innhold\UtenCareCVC\','E:\Master_data\Treningsdata\Dokum
enter\2-forkortelser\Alt\Test\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)',
'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Alt_innhold\UtenCareCVC\','E:\Master_data\Treningsdata\Dokum
enter\2-forkortelser\Alt\Training\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)',
'Trening' );



//Fordele data for Del6, CareCVC

//call Fordeling ('E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Kun_Del6\CareCVC\','E:\Master_data\Treningsdata\Dokumenter\2
-forkortelser\Del6\Test\1-CareCVC\', '$(vFordelingsfil_CareCVC)', 'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Kun_Del6\CareCVC\','E:\Master_data\Treningsdata\Dokumenter\2
-forkortelser\Del6\Training\1-CareCVC\', '$(vFordelingsfil_CareCVC)','Trening' );



//Fordele data for Del6, UtenCareCVC

//call Fordeling ('E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Kun_Del6\UtenCareCVC\','E:\Master_data\Treningsdata\Dokument
er\2-forkortelser\Del6\Test\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)', 'Test'
);

//call Fordeling ('E:\Master_data\Preprosessering\2-
Forkortelser\Dokumenter\Kun_Del6\UtenCareCVC\','E:\Master_data\Treningsdata\Dokument
er\2-forkortelser\Del6\Training\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)',
'Trening' );



//*************************************************************************************
*************************************************************************************
************************************************

//Kommandoer for å fordele STEMMING

//Fordele data for Alt, carecvc

//call Fordeling ('E:\Master_data\Preprosessering\3-
Stemming\Dokumenter\stemming\Alt_innhold\CareCVC\','E:\Master_data\Treningsdata\Doku
menter\3-stemming\Alt\Test\1-CareCVC\', '$(vFordelingsfil_CareCVC)', 'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\3-
Stemming\Dokumenter\stemming\Alt_innhold\CareCVC\','E:\Master_data\Treningsdata\Doku
menter\3-stemming\Alt\Training\1-CareCVC\', '$(vFordelingsfil_CareCVC)','Trening' );
```

```
//Fordele data for Alt, UtenCareCVC

//call Fordeling ('E:\Master_data\Preprosessering\3-
Stemming\Dokumenter\stemming\Alt_innhold\UtenCareCVC\','E:\Master_data\Treningsdata\
Dokumenter\3-stemming\Alt\Test\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)',
'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\3-
Stemming\Dokumenter\stemming\Alt_innhold\UtenCareCVC\','E:\Master_data\Treningsdata\
Dokumenter\3-stemming\Alt\Training\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)',
'Trening' );



//Fordele data for Del6, CareCVC

//call Fordeling ('E:\Master_data\Preprosessering\3-
Stemming\Dokumenter\stemming\Kun_Del6\CareCVC\','E:\Master_data\Treningsdata\Dokumen
ter\3-stemming\Del6\Test\1-CareCVC\', '$(vFordelingsfil_CareCVC)', 'Test' );

//call Fordeling ('E:\Master_data\Preprosessering\3-
Stemming\Dokumenter\stemming\Kun_Del6\CareCVC\','E:\Master_data\Treningsdata\Dokumen
ter\3-stemming\Del6\Training\1-CareCVC\', '$(vFordelingsfil_CareCVC)','Trening' );



//Fordele data for Del6, UtenCareCVC

//call Fordeling ('E:\Master_data\Preprosessering\3-
Stemming\Dokumenter\stemming\Kun_Del6\UtenCareCVC\','E:\Master_data\Treningsdata\Dok
umenter\3-stemming\Del6\Test\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)',
'Test' );

call Fordeling ('E:\Master_data\Preprosessering\3-
Stemming\Dokumenter\stemming\Kun_Del6\UtenCareCVC\','E:\Master_data\Treningsdata\Dok
umenter\3-stemming\Del6\Training\0-UtenCareCVC\', '$(vFordelingsfil_UtenCareCVC)',
'Trening' );
```