TDT4215 — Web-intelligence

Prosjektrapport

Aleksander Gisvold, Jørgen Ekeland, Tri M. Nguyen

April 2013

Contents

1 Introduksjon

Dette er en rapport for prosjektet i faget TDT4215 Web-intelligens ved NTNU, våren 2013. I denne oppgaven er vi blitt bedt om arbeide som et lite konsultentbyrå for å løse en oppgave om websøk og webintelligens. Den norske legemiddelhåndboken er blant annet blitt brukt som datakilde for å løse denne oppgaven. Oppdragsgiveren vil at vi skal forbedre dagens nøkkelordbaserte søkemetoder, og utvikle et søkesystem som benytter seg av pasientjournaler for å bygge et søk i Legemiddelhåndboken.

For å implementere vår løsning har vi benyttet teknologier som Java, Apache Lucene og Apache Jena.

2 Arkitektur

Etter undersøkelse bestemte vi oss for å bygge systemet vårt ved hjelp av en 'rør og filter'-arkitekturen (pipe-and-filter). Det gjorde det lettere for oss å kunne jobbe på filer individuelt, og unngå dobbeltarbeid, ved eks. at resultatene fra en operasjon lagres før vi utførte neste operasjon. Siden pasientjournalene og den norske legemiddelhåndboken kun skulle tolkes en gang, mens søkene skulle utføres gjentatte ganger, med ulike søkeparametere, var dette den mest hensiktsmessige løsningen.

En viktig bit av søk i dokumentsamlinger er å kunne indeksere termer i samlingen av dokumenter. For å gjøre dette, tolket vi ATC- og ICD10-kodene og lagret resultatene. Ved å preprosessere NLH og pasientjournalene, var det mulig for søkemotoren å bygge en spørring basert på termer i pasientjournalen. Denne spørringen ble dermed brukt for å søke i ATC- og ICD10-indeksene. Ved å bruke Java-biblioteket Apache Lucene, implementerte vi en metode for å rangere resultatene basert på sammenhengen mot ATC-/ICD10-kodene.

Merk at javakoden er skrevet på engelsk. Vi valgte engelsk fordi det er en uskrevet standard at man programmerer på engelsk. Det gjorde det også vesentlig enklere å kunne spørre om hjelp via nett.

ATC/ATCParser:
ATC-kodene ble gitt av fagstaben som en .owl-fil. ATC benyttes for å lese en prologfil. Ved å lese filen, kunne 'ATCParseren' separere innholdet og lagre det som separate ATC-objektene for enklere søk.

ICD/ICDParser:
Legemiddelhåndoken er åpent tilgjengelig på nett, som HTML-filer. 'NLHParseren' leste filene sekvensielt og fjernet HTML-etiketter fra filene, for å fjerne støy. Etter å ha fjernet HTML-etiketter ble hvert dokument gjort om til en XML-fil. XML-filene ble deretter lest sekvensielt for å fjerne stoppord og deretter stemt. For å fjerne

stoppord og stemme teksten benyttet vi et åpent tilgjengelig stemmeprogram ved navn 'SnowballStemmer'.

Pasientjournaler:
Pasientjournalene ble gitt som en docx-fil, av fagstabel. Vi delte filen i separate filer for hver pasientjournal. Ved hjelp av javaklassen 'CaseReader', som vi implementerte på egenhånd, fjernet vi stoppord og stemte pasientjournalene.

Indeksering:
Ved hjelp av java-biblioteket Apache Lucene leste vi inn ATC-kodene og indekserte hver etikett og hver kode. Programmet indekserte også hver ICD10-etikett, -kode og synonymer for kodene. Dette ble lagret i separate filer, slik at det ble lettere å aksessere i ettertid.

Søk:
Med pasientjournalene og NLH omgjort til en felles standard, var det vesentlig enklere å gjøre søkearbeid. Søkemotoren tar en setning fra hver pasientjournal og kapittel fra NLH og søker gjennom ATC- og ICD10-kodene for å finne relevante treff. Dette ble gjort ved hjelp av Apache Lucene, for å enklere lese indekseringen og senere rangere treffene.

vectorSpace package:
Vi valgte å benytte 'Vector Space Model' for søk og rangering. Implementasjonen av dette er nærmere beskrevet i Teorier

textitDocumentVector:
DocumentVector-klassen er brukt for å skape vektorer av de indekserte dokumentene. Klassen benytter en 'DocumentCollection'-lytter for å unngå av det oppstår duplikate vektorer.

textitDocumentCollection:
'DocumentCollection'-klassen lager en samling av vektorene produsert av 'DocumentVector'. Dette lagres som et 'HashMap'. Klassen og samlingen oppdateres hver gang et nytt dokument er lagt til i samlingen.

textitVectorSpace:
Denne klassen utfører en beregning for å kalkulere cosinus-likheten mellom vektorer. Mer om hvordan dette beregnes er beskrevet i teorier

textitVectorMain:
Denne klassen består av 'main'-metoden for å kjøre programmet. Denne klassen benytter seg av alle de tidligere beskrevne klassene for å utføre en spørring som en helhet.

3 Begrensninger og krav gitt av oppgaven

Oppgaven ga enkelte begrensninger og krav til vår implementasjon.

1. Autokodet ATC
   (a) Finn relevante ATC-koder bestemt av en sammenheng mellom beskrivelsen av ATC-koden og setningene i pasientjournalene, for hver enkelt setning i journalene.
   (b) For å lagre en sammenheng mellom en setning og en ATC-kode, lag en tabell hvor hver enkelt rad inneholder dokumentnummeret, setningsnummeret etterfulgt av den relevante ATC-koden (rangert).
   (c) Gjør tilsvarende for terapikapitlene i Legemiddelhåndboken. Inkluder kun subkapitler og subsubkapitler.
2. Autokodet ICD-10
   (a) Tilsvarende som gjøres for ATC, skal gjøres med ICD-10, men ICD-10 klassifiseringer skal benyttes istedenfor.
3. Rangering ved bruk av vektormodeller
   (a) Vi skal benytte metoder for informasjonsgjenfinning for å trekke ut og sammenligne semantisk informasjon i dokumenter, basert på statistisk informasjon.
4. Evaluering
   (a) Vi skal skrive en rapport hvor vi presenterer teknikker/metoder for å evaluere resultatene automatisk.
   (b) For hver unike metode, skriv en evaluering
5. Forbedre rangeringen
   (a) We skal kun benutte kodene fra oppgave 1 og 2 for å velge og rangere de relevante kapitlene i Legemiddelhåndboken.
   (b) Vi skal kombinere resulatene fra oppgave 6a med resultatene fra oppgave 3. Vi må også finne en måte å vekte de ulike rangeringene gitt av de ulike fremgangsmåtene.
6. 'Gold Standard'
   (a) Vi skal evaluere våre ulike rangeringsmetoder ved hjelp av en "gylden standard" gitt av fagstaben.
7. Mulige forbedringer
   (a) Vi skal forklare ulike måter å videreutvikle og forbedre systemet på.

4 Teorier

4.1 Teoretisk beskrivelse

En av de sentrale teoriene innen informasjonsgjenfinning er 'Vector Space Model'. Dette er en algebraisk modell, som representerer dokumenter som vektorer av sine indekserte termer. Prosedyren som denne modellen tar i bruk, er delt i tre deler:

1. Dokumentindeksering er å trekke ut termer basert på innholdet.
2. Vekting av de indekserte termene, ved hjelp av forekomstfrekvens for termene
3. Rangering av dokumentene med hensyn til spørringen, vektet av en likhetsmåling

I denne modellen representeres både dokumenter og spørringer ved en vektor, på formen:

$$d_j = (w_{1,j}, w_{2,j}, ..., w_{n,j})$$

$$q = (w_{1,q}, w_{2,q}, ..., w_{t,q})$$

hvor hver enkelt dimensjon representerer en separat term. Begrepet 'term' kan variere noe. En term kan representere en setning, et ord eller en del av en setning. I vår oppgave har vi valgt å håndtere dem som ord, for å gjøre det enklere for oss å indeksere. Ved indeksering velger man alltid å representere kun de termene som er representert i dokumentene. Disse termene har derfor en verdi over null. Verdien er som regel vektet ved hjelp av en funksjon som kalkulerer verdien basert på frekvensen av termen. Den mest brukte funksjonen er tf*idf, 'term-frequency* inverse- document- frequency'. Denne måten å vekte termer på forteller oss hvor viktig et ord er i forhold til dokumentet det er en del av, og i forhold til mengden av dokumenter i samlingen det søkes i. Tf*idf-verdien vil øke proporsjonalt med antallet ganger ordet forekommer i dokumentet, men justeres ved hjelp av antallet forekomster i resten av dokumentene. Dette blir gjort for å ta hensyn til ofte brukte ord, som kan forekomme ofte i alle dokumentene kontra ord som forekommer kun i et spesifikt dokument.

Når dokumentene indekseres, forsøker vi å fjerne ordene som ikke beskriver innholdet. Eksempler er: dette, er, en, et, og, o.l. Disse ordene kalles "stoppord".

Etter å ha vektet dokumentene, vil metoden rangere resultatene ved hjelp av å regne ut "cosinus-likheten" mellom dokumentene og spørringen. Ved å beregne "cosinus-likheten" menes å sammenligne vektoren for spørringen med de ulike vektorene for dokumentene. Dette gjøres ved å kalkulere vinkelen mellom vektorene. Formelen for dette er gitt ved:

$$\cos(\theta) = \frac{d_2 \times q}{\| d_2 \| \times \| q \|}$$

hvor telleren betegner vektorproduktet og nevneren betegner normaliseringen. Fordi alle vektorverdiene er positive og aldri 0, vil et resultat på 0 bety ortogonalitet mellom vektorene og det er intet samsvar mellom dem.

Denne modellen har imidlertid enkelte begrensninger:

- Store dokumenter representeres på en ubalansert måte, med et lite skalarprodukt og høy dimensjon
- Ikke alle relevant ord vil bli matchet. Substringer kan gi 'falske positiver'
- Synonymer kan forstyrre den semantiske tolkningen av dokumentet
- Når dokumenter indekseres blir rekkefølgen av termer tapt.
- Det antas at det er en statistisk uavhengighet mellom ord

4.2 Teoriene i praksis

For å benytte oss av 'VSM' var vi nødt til å gjøre en del forberedende arbeid på dokumentene våre. Pasientjournalene ble gitt som en tekstfil. Det første som måtte gjøres var å fjerne stoppord, for å forbedre indekseringen. Etter fjerningen av stoppord benyttet vi oss av en 'Stemmer' for å stemme alle ordene til sin grunnform. For å utføre stemmingen benyttet vi 'Snowball Stemmer', et program gjort tilgjengelig publisering av kildekoden på nett. Etter å ha blitt stemt var dokumentene klare for å bli søkt i. Teorikapitlene i NLH trengte også preprosessering før det var klart til bruk. Først måtte vi fjerne HTML-merker, for å lettere søke i teksten, for så å gjøre kapitlene om til XML-filer. Teorikapitlene ble også stemt. ICD10-kodene ble publisert som en .owl-fil. For å lettere filtrere, lagde vi klasser for hver enkelt etikett i ICD10 og ATC. Etter å ha laget klasser, benyttet vi oss av Apache sitt 'Digester'-bibliotek for hver enkelt etikett. Til slutt benyttet vi Apache sitt 'Lucene'-bibliotek for å indeksere alle klassene. De indekserte filene ble lagret på disk, for å slippe å gjøre arbeid om igjen. På denne måten var det mulig for oss å benytte teorikapitlene som søkeparametere når vi søkte gjennom de indekserte filene. Som kjent, var målet å finne den beste matchen mellom de pasientjournalene og terapikapitlene i NLH. Som tidligere nevnt, benyttet vi VSM-modellen for å utføre søket og rangeringen av dokumentene. Vi fulgte stegene presentert i den teoretiske beskrivelsen til punkt og prikke. Måten vi gjorde det på var som følger:

- Bygget et array av preprossesserte teorikapitler
- Vektet ordene i dokumentet, ved hjelp av tf*idf-formelen
- Kalkulerte cosinus-likheten, ved hjelp av ovennevnte formel
- Rangerte dokumentene

4.3 Kommentarer til løsningen

Vi valgte å dele NLH i kapitler og subkapitler, da dette var måten HTML-filene var delt inn på. Det eksisterer subsubkapitler innenfor subkapitlene, men subsubkapitlene er ikke separate filer. Vi fant ingen god måte å løse dette problemet på.

5 Resultater

5.1 Top 10 resultater ved benyttelse av VSM, med og uten vekting ved tf*idf

Pos	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8
1	t3.1	t10.2	t1.10	t8.3	t12.10	t1.3	t20.2	t11.3
2	t10.2	t19.5	t1.14	t19.6	t1.6	t11.3	t21.1	t10.11
3	t23.1	t10.1	t19.1	t24	t12.9	t16.9	t22.4	t1.3
4	t14.5	t10.3	t3.1	t8.4	t19.6	t1.11	t20.1	t1.10
5	t15.1	t24.1	t7.9	t8.1	t17.2	t10.2	t19.2	t5.1
6	t21.1	t11.1	t8.1	t15.3	t13.2	t10.3	t17.3	t1.13
7	t19.3	t10.8	t1.2	t10.2	t22.1	t11.4	t14.1	t1.12
8	t8.1	t22.3	t1.11	t19.5	t12.3	t1.10	t6.2	t1.9
9	t19.2	t8.3	t10.3	t12.8	t5.5	t1.7	t6.5	t11.4
10	t16.13	t14.5	t8.8	t8.6	t1.17	t12.3	t22.2	t16.5

Table 1: Resultater ved bruk av VSM(vektet ved tf*idf)

Pos	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8
1	t3.1	t3.3	t1.10	t8.3	t5.5	t2.2	t20.2	t11.3
2	t15.1	t8.9	t15.1	t2.2	t1.10	t2.1	t20.1	t1.7
3	t16.13	t10.2	t2.1	t12.3	t17.2	t8.3	t21.1	t1.16
4	t14.1	t1.7	t6.2	t8.10	t8.9	t12.3	t22.4	t24.1
5	t14.3	t2.2	t8.3	t6.5	t6.5	t8.10	t22.2	t1.10
6	t8.1	t2.1	t1.7	t5.4	t10.2	t5.4	t19.2	t6.2
7	t24.2	t6.5	t16.7	t8.9	t12.3	t12.6	t5.5	t8.9
8	t10.2	t12.3	t4.6	t1.10	t1.7	t17.1	t12.1	t19.6
9	t16.8	t5.4	t3.1	t2.1	t3.1	t3.3	t6.2	t10.2
10	t7.7	t8.3	t21.1	t10.2	t2.2	t8.9	t17.3	t3.3

Table 2: Resultater ved bruk av VSM(uten vekting ved tf*idf)

5.2 Eksempler på ulike resultater med/uten tf*idf

5.2.1 Pasientsak nr 5

Pasienten er en 64 år gammel kvinne som tidligere er stort sett frisk. De siste 5 månedene har hun merket følelse av ufullstendig tømming når hun har avføring. Ved flere anledninger har det vært spor av friskt blod i avføringen. Selv tilskriver hun dette hemorroider som hun har hatt før, og hun har ventet på at det skulle gå over.

Hun har vært hos sin egen lege som ikke finner noe galt ved vanlig undersøkelse. Ved rektaleksplorasjon (kjenne i endetarmsåpningen med finger) kjenner legen så vidt kanten av en uregelmessighet i tarmveggen. Legen ser spor etter gamle ytre hemorroider men ingen sannsynlig blødningskilde nå. Prøver på blod i avføringen er positive. Blodprøver viser en lett blodmangel (hemoglobin 10.8; normalt for kjønn og alder er ¿12). Øvrige blodprøver var normale. Pasienten blir henvist til colonoscopi (endoskopisk undersøkelse av tykktarmen).

5.2.2 Resultater

Uten tf*idf: T5.5 Depresjoner

Med tf*idf: T12.10 Anorektale Forstyrrelser

Her ser vi at uten vekting ved bruk av tf*idf vil resultatet være ubrukelig, da det ikke har noen sammenheng med pasientens symptomer. Uten bruk av tf*idf vil T12.10(det mest nyttige dokumentet) ikke være blant de øverste 15 treffene søket vil returnere. Det er tydelig at tf*idf er essensielt i denne sammenheng.

5.3 Pasientsak nr 6

Ved første konsultasjon ble det funnet forstørrede tonsiller med hvitlig belegg og forstørrede glandler på begge sider av halsen. Det ble tatt halsprøve til strept.test som var positiv. Det fremkom at kjæresten nylig hadde gjennomgått streptokokktonsilitt. Det ble startet behandling med peroral penicillin (Apocillin) i vanlig dosering.

Pasienten kommer til ny konsultasjon etter manglende effekt av behandlingen med penicillin. Han ble verre, fikk mer svelgbesvær, fikk ikke i seg fast føde, og hadde også besværligheter med å få i seg væske. Samtidig var han vedvarende slapp og i dårlig allmenntilstand.

5.3.1 Resultater

Uten tf*idf: T2.2 Legemiddelbehandling av vanlige kreftsykdommer

Med tf*idf: T1.3 Mononukleose

Resultatet uten tf*idf foreslår at pasienten har kreft, noe som kan være en svært forstyrrende diagnose dersom den gis feilaktig. Ved bruk av tf*idf vil Mononukleose dukke opp som et mye bedre resultat. Tf*idf gir en klar forbedring av søkeresultatene.

6 Diskusjon

Det finnes flere viktige aspekter rundt informasjonsgjenfinning. Blant annet vil det ofte være naturlig å se på precision kontra recall. Som en del av kursmaterale fikk vi også med en såkalt 'gullstandard' som skulle brukes for å måle ytelsen av søkemotoren som vi har laget.

Precision går på det å se på antall relevante dokumenter av de dokumentene som er valgt.

Precision = (relevante og hentede dokumenter) / hentede dokumenter. Eks. Om vi har hentet 2 dokumenter, og bare 1 er relevant. Så vil precision være 50%.

Da ser vi mens recall ser på på antall relevanter dokumenter av totale dokumenter. Recall = (relevante og hentede dokumenter) / (totale dokumenter). Eks. om det finnes 20 dokumenter i kolleksjonen, og 10 av dem er hentet, så vil recall være 50%. Precision ville vært 100% i dette tilfelle.

For å skape et bedre bilde av situasjonen, kalkulerte vi den interpolerte presisjonen for fire recall-nivåer: 25%, 50%, 75% and 100%.

Recall	$\overline{P}$(r)
25%	68%
50%	46%
75%	28%
100%	25%

Table 3: Precision vs Recall

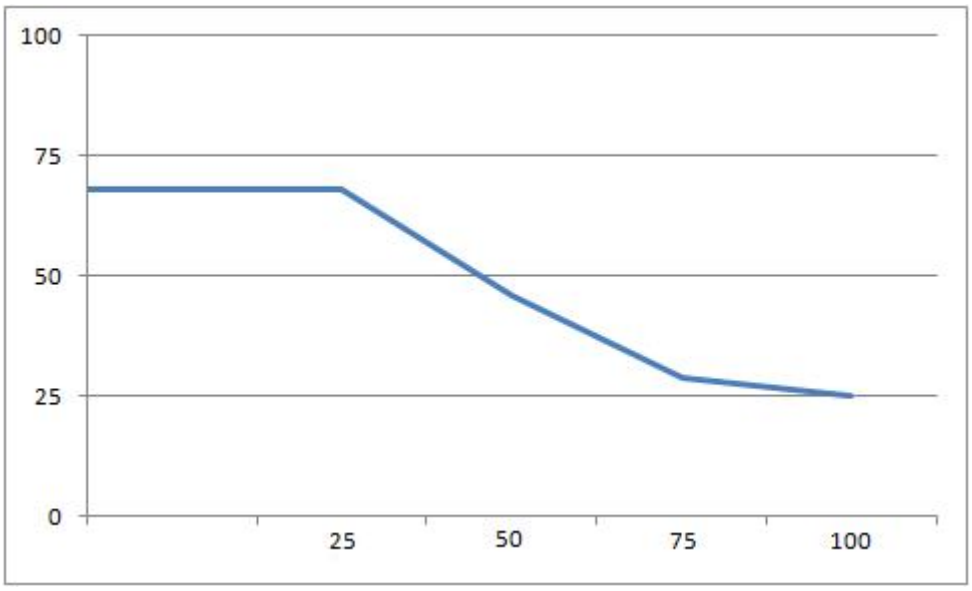

Figure 1: Precision vs Recall

Siden Legemiddelhåndboken er et stort dokument som er videre delt i kapitler, subkapitler og subsubkapitler, bestemte vi oss for å kun håndtere kapitler og subkapitler som separate dokumenter. Bakgrunnen for dette var at kapitler og subkapitler har egne HTML-filer i den originale dokumentsamlingen. Det ville vært en mer optimal løsning å dele videre inn i subsubkapitler, for å få resultater med mindre støy.

Grunnet støy i resultatene fikk vi tidvis dårlig presisjon, spesielt i pasientsak 7 og 8, hvor det ble retunert for mange dokumenter.

Måten vi delte dokumentene opp på førte også til at vi i enkelte tilfelle fikk totalt feilaktige søk. I pasientsak 5 var ikke 'gold standard'-dokumentet i søkeresultatene i det hele tatt.

I pasientsak 6 var det forventede dokumentet det høyst rangerte dokumentet. I resten av pasientsakene var de forventede dokumentene blant de fem høyest rangerte dokumentene.

7 Videreutvikling

Det er mye ved dette prosjektet som kunne vært gjort bedre. Vi ser på noen av aspektene under.

7.1 Forbedringer i implementasjonen

TF og TF-IDF er ikke egnet for å håndtere ulike dokumentformat som HTML, XML og .owl. Dette førte til at vi måtte gjøre en god del implementasjon for å gjøre preprosessering for å få dokumentene på en felles standard. I etterkant har vi lært at det eksisterer måter å gjøre dette på en mer egnet måte. Ved hjelp av 'Semantic-Sensitive Web Information Retrieval Model' kan man enklere hente ned dokumenter av ulikt format.

Vi valgte å skille kapitler i NLH på hovedkapitler og subkapitler. Dette førte til at resultatene våre ved enkelte tilfelle ble suboptimale. NLH deler videre inn i subsubkapitler, selv om disse dokumentene er deler av subkapitlene. Ved tolkningen og prosesseringen av NLH kunne vi delt dokumentene ytterligere. Vi gjorde et forsøk på å gjøre dette, men resultatene våre ble ustabile og dermed oppstod feil. Dette ble for tidkrevende for at vi kunne implementere det.

7.2 Forbedringer i arkitekturen

Koden er generelt dårlig dokumentert og kommentert. Det vil være vanskelig for de som måtte ta over koden å arbeide videre på den.

7.3 Gruppesammensetning

Gruppen vår bestod av tre studenter med tre ulike timeplaner. Dette førte til vanskeligheter med å jobbe sammen. Dette førte til økte komplikasjoner med kommunikasjonen i gruppen. Vi burde arrangert flere økter hvor vi jobbet sammen.

8 Konklusjon

Gjennom prosjektet har vi lært om algoritmer og fremgangsmåter innenfor søk og informasjonsgjenfinning. Vi har sett tydelig sammenheng mellom bruk av algoritmer og hvilke resultater man oppnår ved bruk av ulike algoritmer. Vi har også lært å rangere resultatene og se hvorfor resulatene ble som de ble.

Oppgaven ga oss en god demonstrasjon av hvor viktig det er å kombinere ulike teorier for å oppnå best mulige resultater, selv om dette kan være krevende. Implementasjonen vår lærte oss viktigheten av å skille dokumenter for å gjøre de mest mulig uavhengige. Dokumentsamlinger er ikke alltid laget med en søkemotor i tankene, noe vi lærte når vi lagde vår søkemotor.

Vi konkluderer med at vi har gjort en god innsats på å implementere en søkemotor for et vanskelig domene, og at læringsverdien av dette prosjektet har vært høy.

9 Appendix

Table 4: Norske stoppord

A - D	D - H	H - K	K - N	N - S	S - Å
alle	dit	har	kun	noe	so
andre	ditt	hennar	kunne	noen	som
at	du	henne	kva	noka	somme
av	dykk	hennes	kvar	noko	somt
bare	dykkar	her	kvarhelst	nokon	start
begge	då	hit	kven	nokor	stille
behandling	eg	hjå	kvi	nokre	syk
ble	ein	ho	kvifor	ny	så
blei	eit	hoe	lage	nå	sånn
bli	eitt	honom	lang	når	tid
blir	eller	hos	lege	og	til
blitt	elles	hoss	lik	også	tilbake
bort	en	hossen	like	om	um
bra	ene	hun	man	opp	under
bruk	eneste	hva	mange	oss	upp
bruke	enhver	hvem	me	over	ut
både	enn	hver	med	pasienten	uten
båe	er	hvilke	medan	pasienter	var
bør	et	hvilken	meg	pga	vart
ca	ett	hvis	meget	på	varte
da	etter	hvor	mellom	rett	ved
de	for	hvordan	men	riktig	verdi
deg	fordi	hvorfor	mens	samme	vere
dei	forsøke	i	mer	seg	verte
deim	fra	ikke	mest	selv	vi
deira	fram	ikkje	mg	si	vil
deires	få	ingen	mi	sia	ville
dem	før	ingi	min	sidan	vite
den	først	inkje	mine	siden	vore
denne	gjorde	inn	mitt	sin	vors
der	gjøre	innen	mot	sine	vort
dere	god	inni	mye	sist	vår
deres	gå	ja	mykje	sitt	være
det	går	jeg	må	sjøl	vært
dette	ha	kan	måte	skal	å
di	hadde	kom	ned	skulle	
din	han	korleis	nei	slik	
disse	hans	korso	no	slutt	