Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Andreas Hammer
Dina Rosvoll
Oda Steinland Skaug

Utvikling av en webapplikasjon for å tilgjengeliggjøre analyse av trender i mediebildet, og forskning på norsk sentimentanalyse ved bruk av maskinlæring

Bacheloroppgave i Bachelor i ingeniørfag, data
Veileder: Ole Christian Eidheim, Bjørnar Kjenaas Mælum
Mai 2019

NTNU
Kunnskap for en bedre verden

IBM

Andreas Hammer
Dina Rosvoll
Oda Steinland Skaug

Utvikling av en webapplikasjon for å tilgjengeliggjøre analyse av trender i mediebildet, og forskning på norsk sentimentanalyse ved bruk av maskinlæring

Bacheloroppgave i Bachelor i ingeniørfag, data
Veileder: Ole Christian Eidheim, Bjørnar Kjenaas Mælum
Mai 2019

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Sammendrag av Bacheloroppgaven

Tittel:	**Utvikling av en webapplikasjon for å tilgjengeliggjøre analyse av trender i mediebildet, og forskning på norsk sentimentanalyse ved bruk av maskinlæring**
Oppgave no.	108
Dato:	20.05.2019
Deltakere:	Andreas Hammer Dina Rosvoll Oda Steinland Skaug
Veiledere:	Ole Christian Eidheim Bjørnar Kjenaas Mælum
Oppdragsgiver:	Industrial Business Machines, IBM
Kontaktperson:	Lars Hovind, hovind@no.ibm.com, +47 922 59 690
Nøkkelord:	Systemutvikling, Maskinlæring, Sentimentanalyse
Antall sider:	67
Tilgjengelighet:	Åpen

Sammendrag: Bacheloren består av en todelt oppgave, med fokus på systemutvikling og maskinlæring. Første del består av en webapplikasjon som forenkler og tilgjengeliggjør sentimentanalyse på entiteter i media. Den andre delen består av forskning på ulike maskinlæringsmodeller og datasett for norsk sentimentanalyse.

Summary of Graduate Project

Title:	**Developing a webapplication for analyzing emotions in media and research on norwegian sentiment analysis using machine learning**
Project no.	108
Date:	20.05.2019
Authors:	Andreas Hammer Dina Rosvoll Oda Steinland Skaug
Supervisor:	Ole Christian Eidheim Bjørnar Kjenaas Mælum
Employer:	Industrial Business Machines, IBM
Contact Person:	Lars Hovind, hovind@no.ibm.com, +47 922 59 690
Keywords:	Machine learning, Sentiment analysis, Webdevelopment
Pages:	67
Availability:	Open

Abstract: The bachelor thesis is a two-part project involving webdevelopment and machine learning research. The first part covers the development of a web application presenting sentiment analysis. The second part focuses on norwegian sentiment analysis, covering different machine learning models and datasets.

Forord

Denne bacheloroppgaven, for dataingeniør vår 2019, er gjennomført i samarbeid med Industrial Business Machines (IBM), for Institutt for datateknologi og informatikk (IDI) ved Norges teknisk - naturvitenskapelige universitet (NTNU).

Å utvikle et system med fokus på sentimentanalyse har vært en spennende og lærerik prosess. Oppgaven kom som et forslag under en samtale med medstudent Sindre Toft Nordal, og ble foreslått for IBM som mulig oppgave under planlegging av samarbeid. Ønsket var å utvikle et system for forenkling og tilgjengliggjøring av sentimentanalyse. Forskningsdelen av bacheloroppgaven kom som et utspring av systemutviklingsoppgaven, dette på grunn av stor interesse for faget og kvalitetssikring av IBM Watson til bruk i systemet. Ingen av studentene hadde forkunnskaper innen maskinlæring i forkant av bacheloroppgaven. Forskningsoppgaven har utviklet seg gjennom å gradvis gå dypere inn i maskinlæring og utforske utfordringene som oppstod av å lage en modell for norsk sentimentanalyse. Vi føler vi har lært utrolig mye nytt, i tillegg til å ha utviklet evnene våre videre innen systemutvikling.

Vi vil takke for hjelp av Bjørnar Mælum, vår veileder ved IBM for teknisk hjelp til både forskningen og utviklingen av produktet. Vi ønsker også å takke Lars Hovind, hovedveileder ved IBM for organiseringen av bacheloroppgaven, kontorplass, workshop hos IBM og koordinering av oppgaven. Vi vil til slutt takke vår veileder på NTNU, Ole Christian Eidheim, for ukentlig oppfølging og rådgivning.

Andreas Hammer

Dina Rosvoll

Oda Steinland Skaug

Oppgavetekst

Gruppen skal i samarbeid med IBM lage en webapplikasjon der brukere kan søke på ord eller fraser, legge til ulike filtre, og få utført sentimentanalyse på valgte entiteter i nyhetsbildet. Bruker vil få en komplett analyse av hvordan entiteten er omtalt i mediebildet i et valgt tidsintervall. Ved hjelp av et grensesnitt kan brukeren fokusere på resultatene fremfor prosessen å utføre sentimentanalyse. Resultatene vil presenteres på en forståelig måte, hvor bruker selv velger hva og hvordan dette skal presenteres. Produktet kan markedsføres mot for eksempel forskere som ønsker å studere trender i mediebildet.

Gruppen skal også forske på sentimentanalyse på norsk ved hjelp av maskinlæringsmodeller. De skal utforske ulike norske datasett og bruke tekstklassifiseringsmodellen fastText til å trene på datasettene, og få en modell som kan analysere norske data basert på sentiment. Dersom gruppen får tid skal de også se på andre alternativer til fastText som bruker en mer kompleks løsning.

Sammendrag

Bachelorprosjektet består av en todelt oppgave med en systemutviklingsdel og en forskningsdel. I systemutviklingsdelen ble det utviklet en applikasjon med formål å forenkle og tilgjengeliggjøre sentimentanalyse for å kunne studere trender i mediebildet. Forskningsdelen undersøkte ulike datasett og modeller for norsk sentimentanalyse.

Systemutviklingsdelen ble løst ved å utvikle en webapplikasjon med funksjonalitet knyttet til søk og sentimentanalyse. Gjennom applikasjonen kan en bruker gjennomføre en sentimentanalyse, med kategoriene *Positive, Negative* og *Neutral*, og en toneanalyse, med kategoriene *Joy, Anger, Sadness, Fear* og *Disgust*. Etter utført søk, med valgte filtre og søkeord, vil bruker få en rapport. Denne inkluderer grafer over gjennomsnittlig score på hver tone/sentiment, vise frem nyhetsartikler aktuelle i søket og graf som viser tone/sentiment per dag i tidsintervallet artiklene er hentet fra. Resultatet har blitt et system som er enkelt og intuitivt å bruke, med formål å presentere statistikk på en oversiktlig og hensiktsmessig måte. Forskere som ønsker å studere trender i mediebildet kan nå bruke det som et verktøy.

I forskningsoppgaven benyttet vi ulike norske datasett annotert med sentiment og forsøkte å automatisk annotere et på egen hånd. I tillegg ble ulike modeller for sentimentanalyse utforsket. For å få et estimat på utviklingen av engelsk sentimentanalyse, sammenlignet vi først en ferdig modell med en utrent modell, fastText, på engelske anmeldelser. Deretter brukte vi samme utrente modell på flere norske datasett annotert med sentiment. Til slutt så vi på alternativer til å lage en egen modell og sammenlignet den med fastText.

Innhold

Figurer

Tabeller

Listings

Definisjoner og forkortelser

UU	Universell Utforming
AI	*Artificial Intelligence*, Kunstig intelligens
ANN	*Artificial Neural Network*, Kunstig nevralt nettverk
Annotere	betyr å tilegne kategorier til data
Backend	er tjenester frontenden er avhengig av for å vise informasjon
CI	*Continous Integration*, Kontinuerlig integrasjon
Embedding	er en måte å representere ord som en vektor, hvor lignende ord får lignende vektor
Emoji/Emotikon	er et ideogram for å uttrykke seg på en annen måte enn ved skriftlig tekst
Epoch	er antall ganger en maskinlæringsmodell ser hele datasettet
FastText	er en modell for hurtig trening på mye data. Skrives med liten forbokstav midt i en setning, men stor på starten av en setning
Frontend	er grensesnittet brukeren samhandler med
Hidden layer	Skjult lag, er lag mellom input og output hvor laget utfører en operasjon på signalet
Hyperparametre	er verdier en modell bruker for å trene som settes i forkant av treningen
LSTM	Long-Short Term Memory, er en type nevron i et nevralt nettverk som husker sekvenser av data
Læringsrate	er en type hyperparameter som bestemmer hvor mye vektene i nevronene skal endre seg for hver kjøring
ML	Maskinlæring, er et felt innen AI, på å lære maskiner å bruke statistiske metoder for å finne mønstre i store datamengder
MMI	Menneske Maskin Interaksjon
N-grams	er sekvenser av N ord av en tekst. Bigrams er sekvenser av to ord i en tekst.

NLP	*Natural Language Processing*, er å prosessere menneskelig språk ved hjelp av en maskin
NLU	*Natural Language Understanding*, er en maskins forståelse av menneskelig språk
NoReC	Norwegian Review Corpus, er et norsk datasett
Nøyaktighet	brukes ofte i rapporten og er brukt for det engelske ordet *accuracy*, som er hvor mye data en modell estimerer riktig i et sett
Overfitting	er når en modell er for godt tilpasset datasettet brukt til trening
Oversampling	er handlingen å øke et datasett ved å tilfeldig doble eksempler
Padding/Truncating	er prosessen å legge til eller fjerne elementer fra vektorer med den hensikt å gjøre flere vektorer til lik lengde.
RNN	*Recurrent Neural Network*, er et type nevralt nettverk brukt i sentimentanalyse som brukes i en strøm av data. LSTM er en type RNN
SA	Sentimentanalyse er å analysere en menneskelig tekst etter følelser
SDG	Stochastic Gradient Descent, er en optimaliseringsalgoritme som oppdaterer vektingen i hvert nevron iterativt basert på treningsdata
Sentiment	er et synonym for følelse
Stemming	er en teknikk for å finne stammen til et ord
Tekstklassifisering	er å gi tekst en kategori (klasse) basert på innholdet i teksten
Tokenization	omformer ord til tall hvor tallet refererer til en ordliste
Tweets	er flertallsformen for en tweet og beskriver meldinger publisert på nettsamfunnet Twitter
Underfitting	er når en modell er trent for lite
Undersampling	er handlingen å jevne ut fordeling av klasser i datasett ved å tilfeldig fjerne eksempler fra andre klasser
Veiledet læring	er en type maskinlæring som går ut på å trene en modell med kategoriserte data

1 Introduksjon og relevans

Nyheter og media er en stor del av hverdagen vår. Begivenheter i samfunnet omtales på daglig basis og resultatet av dette kan brukes til videreutvikling av samfunnet. Bedrifter ser ofte på hvordan produktet deres blir omtalt i medier, og gjør endringer basert på dette. Samme gjelder forskere som studerer trender i mediebildet. Trendene blir omtalt som positiv, negativ eller nøytral. Dette defineres som sentiment og kan hentes ut ved hjelp av maskinlæring. Datamaskiner går gjennom store mengder data på kort tid og kan finne trenden ut i fra et helhetlig nyhetsperspektiv. Det kan derfor oppstå etterspørsel etter muligheten til å analysere hele nyhetsbildet til enhver tid på en enkel og oversiktlig måte.

Gruppen har observert at skytjenestene til IBM for sentimentanalyse krever programmeringskunnskaper og kjennskap til prinsipper for maskinlæring for å få analysert data. Dette kan begrense tilgangen for brukere til denne teknologien. Under utførelsen av prosjektet ville IBM at vi skulle benytte oss av deres teknologi til å utvikle et system som forenkler og tilgjengeliggjør sentimentanalyse.

Muligheten for å se utviklingen av trender innen det norske mediebildet kan være ønskelig for bedrifter og forskere i Norge. Dette krever å finne en maskinlæringsmodell som kan analysere norsk språk for å finne sentiment. IBM tilbyr ikke å analysere norske data for sentiment og gruppen vil derfor undersøke hvilke muligheter som finnes for å lage egen maskinlæringsmodell.

1.1 Problemstilling

1. Hvordan utføres sentimentanalyse på norsk og hvilke utfordringer kan dette by på?
    1. Hvilke utfordringer har kjente annoterte norske datasett for sentimentanalyse?
    2. Hvordan presterer ulike modeller på norsk sentimentanalyse?
2. Utvikling av en applikasjon som forenkler og tilgjengeliggjør sentimentanalyse for forskere som ønsker å studere trender i mediebildet.

1.2 Rapportens struktur

Forskningsdelen er integrert inn i rapporten og ligger sammen med systemutviklingen. Rapporten starter med en teoridel som forklarer maskinlæringsprinsipp og teoretisk bakgrunn for produkt- og designvalg. I neste kapittel vil vi forklare hvilke valg vi har tatt for å få gjennomført oppgaven. Resultatdelen viser hvilke resultater vi har oppnådd. Systemutviklingen deles her inn i tre: Vitenskapelige resultater, knyttet til produkt og design, Ingeniørfaglig resultater, knyttet til mål satt i visjonsdokument, og administrative resultater, knyttet til framdriftsplan og Gantt diagram. Maskinlæringen tar for seg nøyaktighetene oppnådd av de ulike modellene og resultatet av det egenproduserte datasettet. I diskusjondelen går vi dypere inn i hvorfor valg ble tatt, konsekvenser av de og resultatenes svakheter. Konklusjonen vil ta for seg hva vi har kommet fram til i lys av problemstillingene, samt videre arbeid.

2 Teori

I dette kapittelet skal vi forklare teorien bak oppgaven. Først tar vi for oss tidligere arbeid, som består av oppgaver som har utforsket liknende problem. Deretter går vi inn på og dekker det meste innen maskinlæring. Denne er etterfulgt av en kort del om virtuelle maskiner. Til slutt forklares teoretisk bakgrunn for produkt- og designvalg.

2.1 Tidligere arbeid

I tidligere arbeid blir to masteroppgaver og tre forskningsrapporter forklart. De tre første har trent maskinlæringsmodeller for sentimentanalyse, to på norsk og en på engelsk. De to siste forklarer bruk av en ferdiglaget modell og ulike valg innenfor en type preprosessering.

2.1.1 Sentiment Analysis of Norwegian Twitter Messages

En masteroppgave i informatikk på NTNU har forsket på mulighetene for å lage et system for å utføre sentimentanalyse på norske tweets. De benyttet seg av Naive Bayes, Support Vector Machines (SVM) og Maximum Entropy for å løse klassifiseringsoppgavene. Oppgaven har brukt NTNU SmartTagger for å oppnå bedre resultater.

Konklusjonen var at SVM presterte best i dette tilfellet og den gjorde det godt på relativt lite informasjon. Ved å legge til flere datasett ble det observert en nedgang i prestasjon, men denne økte med et større og mer tilfeldig datasett. Det forklares at emotikon kan brukes til klassifisering av polaritet. Interjeksjoner og komparative adjektiver kan også være viktig for klassifisering av polaritet. Det fungerte derimot ikke godt å oversette ord for ord i et sentimentleksikon på et annet språk. Enda et merkbart resultat fra rapporten var at med et mindre datasett kunne nøyaktigheten bli mistenksomt høy. [1]

2.1.2 Sentiment Analysis in Norwegian Political News

En annen masteroppgave i informatikk på NTNU forsket på å utvikle en motor for sentimentanalyse i det norsk politiske nyhetsdomenet. Motivasjonen bak oppgaven var mangelen på forskning innen politiske nyhetsartikler på norsk. De produserte, i samarbeid med VG og NRK, et menneskelig annotert sentimentleksikon på 4000 artikler, hvor hver artikkel hadde en gjennomsnittslengde på 27,3 ord. De fikk mest nøytrale annoterte artikler.

Konklusjonen var blant annet at i norske nyhetsartikler prøver journalister og ikke være for polarisert, men å skjule sentimentet fremfor å eksplisitt uttrykke seg. Resultatet av en to-steg binær klassifiserer var 72,9% med positiv og negativ som klasser. [2]

2.1.3 Sentiment Analysis of Twitter Data for Predicting Stock Market Movements

Forskningsrapporten ser på hvordan endringer i aksjekursene i et selskap, om de stiger eller synker, er korrelert med menneskers offentlige meninger uttrykket i tweets om selskapet. Det blir forklart at

de har prøvd tekstrepresentasjonene Word2vec og Ngram, og hvordan de har utført sentimentanalysen.

Rapporten konkluderer med, og viser, at det foreligger en sterk sammenheng mellom endring i aksjekurs for et selskap og meninger uttrykt om selskapet i offentlige tweets. De viser også til at positive nyheter og tweets i sosiale medier om et selskap, oppfordrer folk til å investere i aksjene til et selskap. Som følge av dette vil aksjekursen for gitt selskap øke [3].

2.1.4 Bag of Tricks for Efficient Text Classification

Forskningsrapporten som ble gitt ut i forbindelse med publiseringen av fastText utforskte en enkel og effektiv basis for tekstklassifisering. Rapporten ble gitt ut av Facebook AI Research, samme gruppe som har laget fastText. I rapporten forklarer de at de trener fastText på mer enn 1 billion ord på under 10 minutter.

Rapporten får frem at fastText forsøker å prestere like godt som modeller basert på nevrale nettverk, men på en brøkdel av tiden. Det første eksperimentet de utførte var på sentimentanalyse. FastText ble trent med 10 hidden units, 5 epoch og en læringsrate på mellom 0,05 og 0,5. De oppdaget at å bruke bigrams økte nøyaktigheten med 1-4 %. Ved å øke n-grams opp til 5 økte nøyaktigheten ytterligere. Tidsbruken var mellom 10 til 15 000 ganger mindre enn til de nevrale nettverkene. FastText slår enkelte nevrale nettverk på nøyaktighet på flere av datasettene testet.

FastText sammenlignes med et avansert nevralt nettverk, LSTM, på et IMDb datasett hvor fastText får en nøyaktighet på 45,2% og LSTM får 45,3%. LSTM slo fastText på alle datasettene testet, men med maksimalt 1%. FastText får nøyaktigheten 91,2% på et Amazon datasett, og 94,6% nøyaktighet ved bruk av *bigrams*. [4]

2.1.5 A Comparative Study of Stemming Algorithms

Forskningsrapporten undersøkte ulike metoder for stemming og sammenlignet de på vegne av bruk, fordeler og begrensninger. Forskjellen mellom stemming og lemmatisering er også forklart i rapporten.

Den viser blant annet at Porter stemmeren produserer beste resultat sammenlignet med andre stemmere og har en lavere feilrate. Det er også en lett stemmer algoritme i forhold til andre av lik type. Begrensningene til Porter er at den produserte stammen ikke alltid er et ekte ord og algoritmen har minst 5 steg og 60 regler som gjør den mer tidkrevende. [5]

2.2 Maskinlæring

Maskinlæring (ML) bruker statistiske metoder for å finne mønstre i store datamengder. Det er i hovedsak en metode for å gjøre ressurskrevende kalkulasjoner på en mer effektiv måte enn det mennesker får til. Maskinlæring er et felt innen kunstig intelligens (eng. *Artificial intelligence*, AI) som forsker på å få maskiner til å løse oppgaver slik som mennesker gjør. Maskinlæring deles gjerne inn i *supervised, semi-supervised* og *unsupervised* læring [6].

2.2.1 Veiledet, Uveiledet og Forsterket læring

Veiledet læring (eng. *Supervised learning*) er å trene modellen på data som er tildelt kategorier, kontinuerlige eller diskré, hvor kategoriene ofte er kalt *labels* eller *classes*. Uveiledet læring (eng. *Unsupervised learning*) trener på ukategoriserte data, mens forsterket læring (eng. *Semisupervised learning*) trener på data hvor enkelte er kategorisert. Veiledet læring deles inn i to ulike modeller, regresjon og klassifisering, ut i fra resultatet den produserer. En regresjonsmodell er en modell som estimerer kontinuerlige verdier, som kostnad eller vekt. En klassifiseringsmodell er en modell som estimerer diskrete verdier, som positiv eller negativ, eller om et bilde er av en katt eller en hund. [7]

2.2.2 NLP

Naturlig språkbehandling (eng. *Natural Language Processing*, NLP) er et eget felt innen kunstig intelligens, som kan bruke maskinlæring for å løse ulike oppgaver. Naturlig språkbehandling går ut på å få maskiner til å kunne analysere menneskelig språk, det vil si å trekke ut mening, tema og følelser fra tekstlig data. [8]

Sarkasme i NLP

Et komplisert problem innen NLP er sarkasme. Sarkasme er handlingen å skrive eller si det motsatte av hva en mener. Det er et problem innen NLP og maskinlæring da det blir vanskelig å finne mening, tema og følelser basert på tekst som betyr det motsatte av hva som står. [9]

2.2.3 N-grams

N-grams er et konsept innen naturlig språkprosessering og består en sekvens med N ord.
Eksempel: “Last donut of the night”
unigrams: “last”, “donut”, “of”, “the”, “night”
bigrams: “last donut”, “donut of”, “of the”, “the night”.
N-grams brukes til å finne relasjoner med ord i nærheten, som er en viktig oppgave innen sentimentanalyse. [4]

2.2.4 Tekstklassifisering

Tekstklassifisering er en oppgave innen NLP som setter tekst inn i sine respektive klasser/kategorier, ved å analysere hva setningen handler om. Det er vanligvis en type veiledet læring som trenger kategoriserte data for å trene og kan brukes til å strukturere store mengder ukategoriserte data. Tekstklassifisering er ikke bare et felt innen maskinlæring, det er også en statistisk tilnærming der det settes definerte regler for kategoriene. En metode er å definere en liste av ord som for eksempel handler om sport, og dersom teksten inneholder en av disse ordene så får teksten kategorien sport. [10]

2.2.5 Sentimentanalyse

Sentimentanalyse (SA) er en del av tekstklassifisering som brukes til å finne følelser, også kalt sentiment, i menneskelig språk. Dette kan enten være positiv, negativ eller nøytral, eller mer avansert slik som glad, sint, trist, overrasket etc. Et alternativ til maskinlæringstilnærmingen er leksikonbasert sentimentanalyse der man ser på hvert ord og finner ut om ordet er positivt eller negativt. Om det er flest negative ord i setningen klassifiseres den som negativ og så omvendt for flest positive. Dersom det er like mange positive og negative ord i teksten, klassifieres den som nøytral. [11]

2.2.6 Nevrale nettverk

Kunstig nevrale nettverk (eng. *Artificial neural networks*, ANN) er en samlebetegnelse på datastrukturer av algoritmer, inspirert av funksjonaliteten til nerveceller i hjernen. Et nevralt nettverk består av flere noder/nevroner som kan være gruppert i flere lag, både synlige og skjulte lag. Skjulte lag (eng. *hidden layers*) er lag mellom input og output hvor nevronene tar en vektet input og produserer en output basert på aktiveringsfunksjonene [12].

Det nevrale nettverket er et sett med tilkoblede inngangs- og utgangsenheter der hver forbindelse har en vekt forbundet med den. Nettverket lærer ved å justere vektene for å forutsi riktig klasse for de oppgitte inngangene [13]. De nevrale nettverkene som brukes kan hovedsakelig deles inn i tre underkategorier: *Multilayer Perceptron*, *Convolutional Neural Networks* og *Recurrent Neural Networks*.

Recurrent Neural Networks (RNN) er utviklet for å behandle en strøm av data. RNN brukes ofte i tekstklassifisering. Vi har to typer noder som ofte blir brukt i RNN og det er GRU (eng. *Gated Recurrent Unit*) og LSTM (eng. *Long-Short Term Memory*). Et lag med *dropout* brukes i nevrale nettverk for å redusere overtilpasning til treningssettet. Det som er spesielt med RNN er at outputen til en node kan sendes tilbake som input i samme node, derav navnet *recurrent* oversatt "tilbakevendende". *Multilayer Perceptron* (MLP) er generell basis for nevrale nettverk og Convolutional Neural Networks (CNN) er utviklet for å brukes i bildegjenkjenning. [14]

Deep Learning

Dyp læring (eng. *Deep Learning*) er en maskinlæringsteknikk som bruker mer enn ett usynlig lag i modellen. For å forstå dyp læring, se for deg et barn som lærer seg ordet hund. Barnet lærer hva en hund er (og hva som ikke er en hund) ved å peke på objekter og si ordet hund. Forelderen sier 'Ja, det er en hund' eller 'Nei, det er ikke en hund'. Etterhvert som barnet fortsetter å peke på objekter, blir han mer klar over hvilken karakteristikk en hund har. Hva barnet gjør, uten å være klar over det, er å klargjøre en kompleks abstraksjon (begrepet hund) ved å bygge et hierarki der hvert nivå av abstraksjon er opprettet med kunnskap som ble oppnådd fra det foregående laget i hierarkiet [15].

2.2.7 Overfitting og Underfitting

Overfitting er når maskinlæringsmodellen er for godt tilpasset treningssettet ved at det lærer seg treningssettets detaljer og støy. Problemet oppstår når modellen skal brukes på nye data uten samme støy, som resulterer i dårlig generalisering. *Underfitting* er når modellen er for dårlig tilpasset treningssettet og det presterer dårlig på andre data. Dette kan eksempelvis forekomme om modellen trener for lite eller ikke har nok frihetsgrader. Det vi ønsker å oppnå er en generalisert modell som er trent godt nok, men ikke for spesifikk på treningssettet. [16]

2.2.8 Trene, Validere og Teste

Vi deler et datasett opp i et treningssett, testsett og et valideringssett. Treningssettet brukes til å trene modellen. Valideringssettet brukes til å validere modellen under trening, spesielt når modellen er trent flere ganger med forskjellig parametere og vi vil se hvilken modell som har best nøyaktighet. Testsettet brukes helt til slutt og gir en offisiell nøyaktighet på modellen. Det er vanlig å bruke en fordeling på 80/20, der 80% av datasettet blir satt av til trening og 20% til test. [17]

2.2.9 Preprosessering

Preprosessering er en teknikk som involverer å transformere rådata til et format som datamaskinen forstår. Det kan også gjøre datasettet mer konsistent som kan gi bedre nøyaktighet. [18]

2.2.10 Stemming

Stemming er en prosess der en fjerner endelsen fra et ord, for så å sitte igjen med ordets stamme. Det er to hovedteknikker for å redusere et ord til sin rot form; stemming og lemmatisering. Forskjellen på disse er at stemming bruker et gitt sett med regler for det gitte språket for å fjerne endelsen. For eksempel, de ulike bøyningsformene av ordet modell blir alle trukket sammen slik: modell, modellen, modeller, modellene - modell. Lemmatisering er en mer kompleks prosess som tar høyde for konteksten ordet er plassert i, samt hvilken ordklasse ordet tilhører for å gi et mer presist resultat. [19]

Stemming har mange bruksområder, der èn er søkemotorer. Ettersom når et søk uføres anser man ikke den spesifikke endelsen på ordet som en viktig del av søket. Stemming blir da brukt for å rense opp i søket slik at forståelsen av hva som søkes på skal bli bedre. [20]

2.2.11 Tokenization

En maskinlæringsmodell kan bare lese tall, ikke tekst. En må derfor gjøre om alle ordene i teksten til tall som henviser til en ordbok. Hvert eksempel i datasettet blir omgjort til en vektor, eller sekvens, som inneholder ordene representert som tall. Et eksempel på hvordan det fungerer: "Det var en spennende dag for studentene" kan bli gjort om til [11, 9, 3, 112, 45, 5, 223]. I noen tilfeller kan tallets størrelse indikere hvor ofte ordet forekommer i datasettet. Ordlisten kan begrenses til å kun inneholde de X mest brukte ordene i datasettet for å minimere treningstid til modellen. [21]

2.2.12 Padding og Truncating

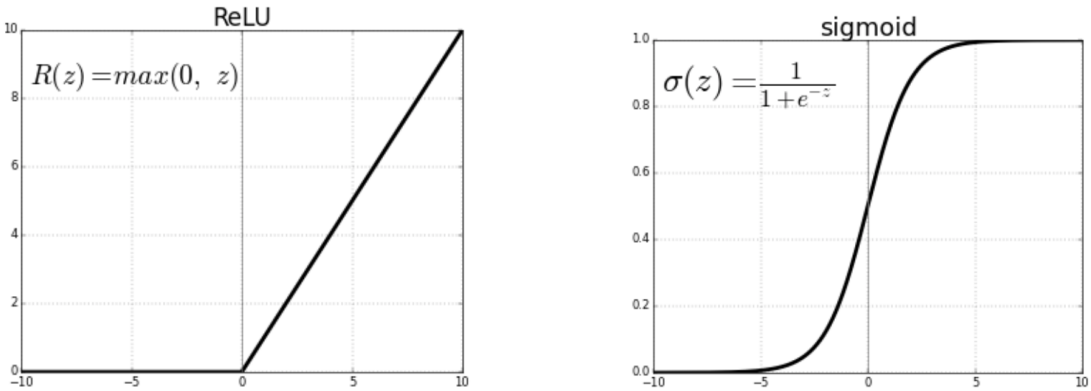
Mange modeller trenger input av en fast lengde. Derfor må vi først finne hvor mange ord/tokens setningene i datasettet består av, og deretter finne gjennomsnittsverdi og makslengde. Vi kan velge om vi skal gjøre omformingen foran eller bak sekvensen. Dersom vi velger foran, og har valgt størrelsen 10 for sekvensene, blir eksempelet over til: [0, 0, 0, 11, 9, 3, 112, 45, 5, 223]. Dette eksempelet var kortere enn størrelsen 10 og vi måtte derfor bruke *padding*. *Truncating* fjerner de første ordene slik at størrelsen blir lik de andre. Det betyr at med størrelsen 10 vil vi kun sitte igjen med de 10 siste ordene i lange setninger. [21]

2.2.13 Embedding

Embedding er en representasjon av tekst hvor ord med lik mening har lignende representasjon. Et embedding lag komprimerer ordforrådet til modellen og gjør det om til en *embedded* vektor. [22]

2.2.14 Aktiveringsfunksjon

En aktiveringsfunksjon er en operasjon som blir utført på signalet gjennom et nevron. Hvert eneste nevron i det nevrale nettverket utfører aktiveringsfunksjonen på input og sender det til output. Vanligvis har hvert nevron samme aktiveringsfunksjon, men det kan variere. Noen kjente aktiveringsfunksjoner er ReLU, sigmoid og softmax. ReLU står for Rectified Linear Unit og er den mest populære aktiveringsfunksjonen brukt i dag. Den gjør om alle negative verdier til 0 (Se figur 2.2.14). Sigmoid gir resultater mellom 0 og 1 og brukes derfor ofte til å estimere verdier hvor output er sannsynlighet (Se figur 2.2.14) [23]. Softmax brukes som aktiveringsfunksjon når vi har flere kategorier som resultat og vi ønsker å se fordelingen av sannsynlighet mellom de ulike kategoriene. [7]



Figur 1: ReLU og Sigmoid funksjon fra Towards Data Science [23]

2.2.15 Optimaliseringsalgoritme

En optimaliseringsalgoritme (eng. *Optimalization algorithm*) brukes for å trene modeller og kan forbedre treningstid og nøyaktighet. Adam algoritmen er en utvidelse av *Stochastic Gradient Descent* (SDG) som oppdaterer vektingen i hvert nevron iterativt basert på treningsdata. SGD bruker samme læringsrate for alle vektene, mens Adam bruker individuelt tilpassede læringsrater til hver vekting. [24]

2.2.16 Loss

Loss er hvor mye modellen avviker fra gjennomsnittet av dataene. Det vil si hvor mye den bommet på å gi ut riktig label, estimere, under treningen. En modell vil under treningen virke inn på lossfunksjonen for å minimere denne så mye som mulig. Det finnes ulike metoder for å beregne loss. Mean Squared Error, Categorical Cross-Entropy, Sparse Categorical Cross-Entropy er eksempler på ulike lossfunksjoner. Mean Squared Error brukes til å regne ut loss ved å ta gjennomsnittet av alle kvadratene av avvikene til dataene. Categorical Cross-Entropy brukes sammen med aktiveringsfunksjonen Softmax hvor kun en klasse kan være korrekt og returnerer svaret som en vektor av nuller, med en 1 for kategorien (kalt "*One Hot Encoding*"). Sparse Categorical Cross-Entropy returnerer en integer og brukes hvor kategori er en tallverdi (1, 2, 3...). Å prøve å minimalisere loss kalles på engelsk *empirical risk minimization*. [21]

2.2.17 Confusion Matrix

Confusion Matrix er en tabell som viser hvilke kategorier modellen fungerer bra og hvilke dårlig.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figur 2: Confusion Matrix fra Towards Data Science artikkel [25]

Den kan brukes til å regne ut *Recall*, *Precision* og *F1* som gir tydeligere tegn til styrker og svakheter ved kategoriene, i motsetning til målingen *Accuracy*. *Accuracy* er antall riktig estimert i forhold til totalt antall estimert.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Precision forteller oss *Accuracy* til en gitt klasse. Den er viktig å undersøke når feilestimering til en gitt klasse har mye å si. Et eksempel er dersom en modell feilvurderer og sender en ikke-kriminell til fengsel. Det vil gi store konsekvenser å estimere feil i klassen kriminell.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Recall gir oss forholdet mellom antall estimert til en gitt klasse og totalt antall estimert av den klassen. *Recall* er viktig å undersøke dersom det er nødvendig å estimere alle innen denne klassen riktig. La oss ta et eksempler hvor man velger ut 100 personer på en flyplass og èn av de er terrorist. Dersom modellen sier at ingen er terrorister, vil *accuracy* bli 99%, men modellen oppdager ikke den virkelige trusselen. Det er i dette tilfellet viktig å estimere klassen terrorist. *F1* score gir et forhold mellom *Precision* og *Recall* og brukes dersom det er ujevn klassefordeling. [25]

2.2.18 Hyperparametre

Hyperparametre er verdier man setter før modellen skal trene og gjør at modellen kan tilpasses datasettet. Verdier som modellen endrer under kjøring kalles parametre. To viktige hyperparametre er læringsrate og antall epoch. Læringsraten bestemmer hvor mye vektene i nevronene skal endre seg for hver oppdatering. Epoch er antall ganger modellen får se hele datasettet. [21]

Batch size er hvor stor del av dataene som skal prosesseres samtidig. Økning av *batch size* fører til at modellen trener raskere, men da må maskinen den kjører på ha nok minne til å kjøre store blokker på en gang. Modellen oppdaterer vektene i funksjonene sine for hver batch. [21]

2.3 Virtuelle maskiner

Universitetet i Bergens side for IT-hjelp definerer virtuelle maskiner som: "En virtuell maskin er en programvare-simulering av en komplett datamaskin som utfører programmer på samme måte som en fysisk datamaskin." [26]

2.4 Systemutvikling

Dette delkapittelet tar først for seg konsepter og verktøy i systemutvikling, og deretter ulike designprinsipp brukt under utformingen av applikasjonen.

2.4.1 Kontinuerlig Integrasjon

Kontinuerlig integrasjon (CI, eng. *Continuous Integration*) er en metode der utviklerens kode legges ut på et felles eksternt lagringsområde. Ved utførelse av kontinuerlig integrasjon vil koden automatisk bygges enten ved innsjekk i et versjonskontrollsystem, som for eksempel Git, eller periodisk [27]. Etter innsjekk, under bygging, vil koden bli verifisert og tester vil bli kjørt. Dersom det oppstår en feil og en av testene ikke blir godkjent, kan vedkommende som sjekket inn koden få varsel om dette og få mulighet til å gjøre nødvendige endringer for at byggingen skal bli godkjent. Dette sikrer at all kode som sjekkes inn er kjørbar og i en potensielt utgivbar tilstand [27]. Ved å kontinuerlig sjekke inn kode vil man kunne oppdage feil tidligere i utviklingsprosessen.

2.4.2 Digitale samhandlingsverktøy

Gjennom digital samhandling har grupper muligheten til å arbeide på fellesdokumenter over internett. For å oppnå dette finnes det flere ulike nettbaserte verktøy, innenfor ulike fagområder og fagfelt, for å effektivisere digital samhandling. Digitale samhandlingsverktøy gjør det enklere for grupper å jobbe sammen for å oppnå ønsket resultat uten overlapping i arbeid. Det er avgjørende for å oppnå god samhandling over nett og er med på å gjøre det enklere for hver enkelt person å henge med og forstå hva som blir gjort innad i et prosjekt. [28]

2.4.3 Versjonskontroll

Versjonskontrollsystemer blir ofte brukt av utviklere når et nytt system skal lages. Det er et system der en enkelt kan oppdatere filer, se tilbake på hva som har blitt gjort tidligere og holde kontroll på endringer i en fil. En kan enkelt gå tilbake til tidligere versjoner av en fil eller finne tilbake til filer dersom en skulle slette eller miste en. [29]

2.4.4 Skytjenester

Å kunne levere tjenester som lagring, database, programvare og servere, over internett er det vi kaller skytjenester. Det er en samlebetegnelse på å få tjenester tilgjengelig fra eksterne servere tilknyttet internett [30]. Dette er med på å gjøre det billigere, den enkelte bruker vil ikke lengre ha behov for selv å kjøpe inn, drifte og vedlikeholde visse hardware og software. Skytjenester må oppfylle strenge krav til sikkerhet, i tillegg vil løsningen kunne være raskere og bedre for den enkelte bruker.

2.4.5 Konseptuell Modell

Konseptuell modell er en måte å vise fram og beskrive fysiske aspekter ved et system på en abstrakt måte [31]. Det er ikke en modell som skal vise grafisk brukergrensesnitt, objektorientert arkitektur eller lignende for et system. Den representerer konsepter og forholdene mellom disse i et problemdomene. Hensikten er å klargjøre betydningen av tvetydige begreper og sørge for at problemer, som kan oppstå, med ulike tolkninger av begreper og konsepter ikke kan forekomme.

2.4.6 Menneske Maskin Interaksjon (MMI)

Menneske Maskin Interaksjon (MMI, eng. *Human Computer Interaction*), er et multidisiplinært fagfelt som forsker og fokuserer på samhandling mellom menneske og maskin [32]. MMI består av flere designprinsipper for utforming av et grafisk brukergrensesnitt og i dette avsnittet vil vi nevne noen av de som har stått sentralt under utformingen av vårt produkt.

Don Norman's Principles of Interaction Design

Prinsippene under er hentet fra Don Normans bok *The Design of Everyday Things* [33] som omhandler hvordan design spiller en viktig rolle i samhandlingen mellom menneske og maskin, og hvordan de ulike partene kan kommunisere med hverandre på best mulig vis.

1. *Affordance* - Tilbydelse
   Hva kan påvirke måten en bruker oppfatter og tenker å bruke systemet på. Hvilke handlinger er mulig for en bruker å utføre og hvordan skal brukeren oppdage disse.
2. *Feedback* - Tilbakemelding
   En handling utført av bruker skal føre til en tilbakemelding. Bruker skal alltid bli informert dersom en handling vedkommende har gjort gir resultater.
3. *Visibility* - Synlighet
   Bruker skal alltid ha oversikt over mulighetene systemet tilbyr, hvilke handlinger han eller hun kan utføre. Bruker skal også, til en hver tid, ha oversikt over tilstanden til systemet.
4. *Consistency* - Konsistens
   Måten en bruker er vant til å utføre en handling i et annet system, skal være lik i et nytt system. Like handlinger, som å logge seg inn eller gjennomføre et søk, skal utføres på samme måte.
5. *Constraints* - Begrensninger
   Ved å sette begrensninger vil en kunne styre handlingene til en bruker og gjøre det enklere for de å tolke hvordan systemet kan brukes. Det hjelper de inn på rett vei til å utføre en handling riktig.
6. *Mapping* - Tilordning
   Forholdet, sammenhengen, mellom ulike komponenter og hva de brukes til. Ved god tilordning vil det være enklere for bruker å skjønne hvordan systemet fungerer og hvordan det skal brukes.

Ben Shneiderman Eight Golden Rules of Interface Design

Ben Shneidermam sine åtte gyldne regler for design av grensesnitt er hentet fra boken *Designing the User Interface* [34] som handler om å designe et velfungerende brukergrensesnitt. Den fokuserer på strategier for å forbedre og effektivisere menneske maskin interaksjon, og diskuterer ulike praktiske teknikker og retningslinjer for å oppnå dette [34].

1. ***Strive for consistency* - Strebe etter konsistens**
   Konsistens bør kreves i ulike deler av systemet, som i menyer, farger, layout, skrifttype, osv. En skal jobbe for å oppnå konsistens i designet til brukergrensesnittet.
2. ***Seek Universal Usability* - Søk Universell brukervennlighet**
   En skal jobbe for universell brukervennlighet, og kjenne til at brukere har forskjellige behov. En skal kunne bruke systemet uavhengig av alder, teknologikjennskap, osv.

3. ***Offer informative feedback* - Tilby informativ tilbakemelding**
   For hver handling utført av bruker i systemet, bør det tilbys en informativ tilbakemelding. En bruker skal alltid bli informert om at handlingen deres har blitt registrert og ført til en endring.
4. ***Design dialogs to yield closure* - Design dialoger med en klar definert endelse**
   Alle handlinger skal ha en klar start, midtdel og avslutning. Enhver handling skal ha en klar avslutning der bruker vil få en informativ tilbakemelding om at handlingen er avsluttet og følgende er resultatet. For eksempel, ved registrering av ny bruker, skal vedkommende få klar beskjed når alle felter er fylt ut og profilen er opprettet.
5. ***Prevent errors* - Forhindre feil**
   Sørg for å lage et *interface* slik at bruker ikke kan gjøre alvorlige feil. Prøv på best mulig måte å designe et system som forhindrer brukeren fra å gjøre feil. Dersom brukeren presterer å gjøre feil, skal systemet informere om en rask og enkel måte å fikse opp i dette som brukeren forstår.
6. ***Permit easy reversal of actions* - Tillat enkel reversering av handlinger**
   Alle handlinger en bruker kan utføre, bør i størst mulig grad være mulig å reversere. Dersom en bruker angrer på et klikk, bør det være enkelt for vedkommende å gå tilbake.
7. ***Keep users in control* - Hold brukerne i kontroll**
   En bruker skal alltid føle på at en har kontroll over systemet. En skal vite hvordan en bruker det og navigerer seg rundt, uansett om vedkommende er en erfaren eller uerfaren bruker.
8. ***Reduce short-term memory load* - Reduser kortsiktig minnebelastning**
   Systemet skal være designet slik at en bruker ikke må huske på informasjon, for eksempel ved sideskifte. Det skal være enkelt å bruke og informativt. Hver side skal inneholde all informasjon som er nødvendig for å kunne bruke den.

2.4.7 Universal Utforming

Universal Utforming (UU) handler om å utforme løsninger slik at de kan benyttes av flest mulig. Det stilles flere krav til hvordan offentlige og private virksomheter skal utforme sine nettsider. Det finnes ingen standarder som løser alle problemer knyttet til universell utforming, og utviklere må ofte være kreative [35].

2.4.8 Informasjonsvisualisering

Informasjonsvisualisering går ut på hvordan en på best mulig måte kan presentere data grafisk. Det er, i følge Waralak V. Siricharoen, en grafisk visuell representasjon av informasjon, data eller kunnskap som er ment å tydeliggjøre og integrere informasjon raskt og tydelig for bruker [36]. Målet med informasjonsvisualisering er å representere abstrakt data på en enkel og oversiktlig måte for å hjelpe bruker med å fordøye og hente innsikt fra de på en effektiv måte. Dette kan gjøres ved hjelp av kakediagrammer, linjediagrammer, grafer og illustrasjoner. [37]

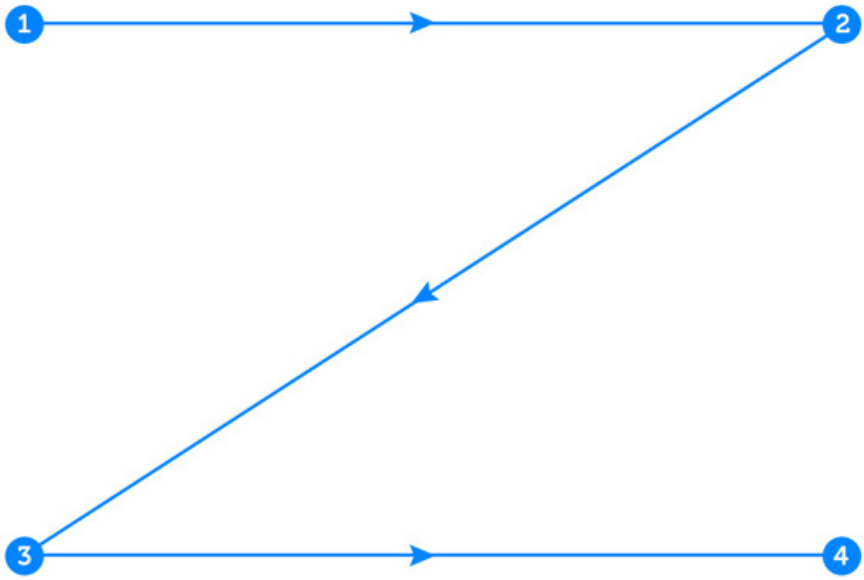
2.4.9 Visuelt hierarki

Visuelt hierarki baserer seg på hvordan en bruker ser på et design. Når en bruker ser et design er det to kognitive prosesser en benytter seg av: Søking og skanning. Søking referer til seerens forsøk på å finne et inngangspunkt på siden, mens skanning referer til seerens oppførelse etter å ha funnet inngangspunkt. Hva inngangspunktet blir kan bli påvirket av farger, størrelse, tekststørrelse og font, visualiseringer og lokasjon. Plassering av komponenter og utforming av design er derfor viktig for

at seerens inngangspunkt og skanning av informasjon rundt dette skal bli på det designer ønsker å formidle. [38]

Z-formet lesemønster

Bruker skanner brukergrensesnittet i en Z-form, der inngangspunktet blir på toppen i venstre hjørnet. Bruker starter i venstre hjørnet og skanner bortover mot høyre. En skanner så videre på tvers av siden fra høyre hjørnet ned til venstre hjørne i bunn [39]. Se illustrasjon under.



Figur 3: Illustrasjon av Z-formet lesemønster. Figuren er hentet fra Instapage [40]

3 Valg av teknologi og metode

Dette kapitlet tar for seg valg av teknologi og metode. Første del beskriver hvordan man kan gjenskape resultatene i maskinlæringen og begrunnelse for valg av datasett, modeller og parametre. Andre del informerer om hvilke teknologier og metoder som ble benyttet i systemutviklingen. Vi skriver også i dette kapitlet om utviklingsmetodikken og rollefordelingen innad i teamet.

3.1 Engelsk sentimentanalyse

Denne seksjonen forklarer hvordan man kan bruke IBM Watson og fastTexts Text Classifier til engelsk sentimentanalyse i Python. Vi trener først en modell i fastText med Amazon anmeldelser og deretter en ny modell med IMDb anmeldelser. Alle modellene blir testet med et utsnitt fra Amazon testsettet.

3.1.1 IBM Watson NLU

Det ble valgt å benytte seg av IBM Watson NLU (*Natural Language Understanding*) ettersom vi bruker denne teknologien i produktet vårt. Vi ønsket å undersøke hvor godt modellen presterer for å kvalitetssikre bruken av det i produktet. IBM Watson NLU tilbyr en mulighet for å analysere sentiment på en tekst, samt sentiment på enkelte entiteter i teksten.

For å bruke IBM Watson NLU kreves det at det opprettes en IBM Cloud NLU-service. Opprettingen avgir en nøkkel (*apikey*) og en URL. Vi bruker programmeringsspråket Python for å teste modellen og installerer "*watson-developer-cloud*", et bibliotek for aksess til IBM Cloud. Kodesnutt 3.1 viser hvordan man aksesserer servicen.

Listing 3.1: Aksessere Watson NLP service

```
nlu = NaturalLanguageUnderstandingV1(
    version='2018-11-16',
    iam_apikey={api_key},
    url='https://gateway-lon.watsonplatform.net/
natural-language-understanding/api'
)
```

For å analysere et datasett må vi gjøre ett kall per linje ettersom funksjonen ikke støtter å sende inn en fil eller tabell av tekst. Hvordan kallet utføres vises i kodesnutt 3.2. Videre forklaring på hvordan man bruker *watson-developer-cloud* NLU er beskrevet i dokumentasjonen til IBM Watson. [41]

NLU Item

IBM Watson bruker *NLU Items* til å sette begrensinger hos brukeren. I gratisversjonen til IBM Cloud har man tilgang på 30 000 *NLU Items* per måned. Et *NLU Item* er basert på antall dataenheter og features som brukes. En dataenhet er definert som 10000 tegn eller færre. Dersom man sender inn en tekst med 10001 tegn bruker man 2 *NLU Items*.

Listing 3.2: Analysere tekst med NLP

```
response = nlu.analyze(
    text=analyse_tekst,
    features=Features(sentiment=SentimentOptions()),
    language='en').get_result()
score = response["sentiment"]["document"]["score"]
```

Amazon Reviews for Sentiment Analysis

Amazon datasettet er et preprosessert engelsk datasett for sentimentanalyse i fastText laget av Adam Bittlingmayer på Kaggle. Datasettet består av totalt 4 millioner anmeldelser klassifisert som enten positiv eller negativ, med en jevn fordeling mellom kategoriene. Det er annotert slik at negativ kategori er representert med 1 og består av anmeldelsene som har fått 1 eller 2 stjerner. Positiv kategori er representert med 2 og er anmeldelsene med 5 eller 6 stjerner. [42]

Eksempel på negativt sentiment: *"__label__1 very poor quality: [...] well all the padding has completely broken down and flaked all over!!!! [...] must purchase a new seat because i will not put her in this one!!!!!!!!!do not purchase!!!!! who only knows what the material is made of(it was made in Italy)less then 1 star"*

Eksempel på positivt sentiment: *"__label__2 The greatest war film in history: [...] This is easily the best Vietnam film and the greatest war film ever. [...] although it leaves the story, themes and characters relatively intact. [...] A classic of the now outdated genre of Vietnam films."* *

Et testsett med 30 000 eksempler ble hentet ut og lagt i en egen fil. Testsettet besto originalt av 400 000 eksempler, men på grunn av maks antall *NLU Items* i IBM Watson måtte testsettet forminskes.

Watson NLU på testsettet

Vi lagde et skript i Python som åpnet filen med testeksemplene og sendte tekstdelen inn ved hjelp av funksjonen beskrevet i kodesnutt 3.2. Etter analysen lagret vi resultatet i en egen fil. Dersom verdien *score* i responsen fra Watson var over 0 klassifiseres resultatet som positivt, og motsatt. For å finne nøyaktigheten sammenlignet vi verdien i resultatfilen med "fasiten" i testsettet.

3.1.2 FastText tekstklassifisering

Vi valgte å bruke fastTexts Text Classifier, av Facebook AI Research group (FAIR), fordi den er kjent for enkel og rask trening av modell. FastText er en utvidelse av Word2Vec modellen *Continous Bag of Words* (CBOW), som prøver å forutse et ord på bakgrunn av ord i konteksten den står i [43]. Modellen trener en lineær klassifisering som faktoriserer små matriser. Den er samtidig trent asynkront på forskjellige CPU'er som bruker *Stochastic Gradient Descent* (SDG, forklart i 2.2.15) og en lineær forminskende læringsrate. Det betyr at du kan kjøre samme parametre med samme datasett og få ulik nøyaktighet på modellen for hver kjøring. FastText la også til *Bag Of N-gram* features for å få med sekvensen i setningene ettersom CBOW ikke tar dette i betraktning. [4]

Fasttext er god på store datasett hvor nøyaktighet på hver enkelt eksempel ikke har mye å si. Den går gjennom hvert eksempel unøyaktig, men uten å bruke mye tid. Eksemplene i datasettet må starte

*Eksemplene er kuttet for å ikke ta for mye plass

med __label__ etterfulgt av kategorien. Selve teksten skal plasseres et mellomrom etter kategorien. Hvert eksempel skilles med en ny linje. Figur 4 forklarer formatet. [44]

```
1   __label__1 flott og kjent metode på å tvinge frem en enighet rbk og kåre måtte i rette
2   __label__1 spissen bør inn foran mål på cornere tenker jeg så skrur den rett i goal
3   __label__0 protip ikke åpne den døra med cooks consensus og resetts kopi av den crappy
4   __label__2 that 70 s band som et særdeles fruktbart samarbeid mellom black sabbath the
5   __label__0 young for faen kjenn din besøkstid du er en skam for drakta 2pl
6   __label__2 stabil korrekt
7   __label__2 &lt&lt en gang for lenge siden da bybanen var på trappene her i bergen skre
8   __label__1 folk som aldri har løst en differensialligning burde ikke ha lov til å mene
9   __label__0 kommenterer ikke annet enn at det er matematisk feil å operere med så
10  __label__1 her er det nokon som slit med problemstillinga
11  __label__0 herregud har du ikke sett harry potter
```

Figur 4: Hvordan datasettet må være formatert for fastText

Hyperparametre

Hyperparametre er verdiene vi fastsetter før modellen skal trene. FastText tilbyr oss å endre hyperparameterne nevnt nedenfor.

- **Epoch**: Antall ganger modellen får se hele treningssettet (iterasjoner).
- **Learning Rate**: Hvor mye modellen endrer seg for hver oppdatering. I fastText er denne en verdi som de forminsker under treningen.
- **WordNgrams**: Hvor mange ord i nærheten av hvert ord det skal forbindes med. I sentimentanalyse er dette viktig da ord kan forsterkes eller få motsatt betydning.
- **MinCount**: Antall ganger et ord må forekomme i treningssettet før det blir med i modellens ordforråd (eng. *Vocabulary*).
- **Dim**: Dimensjonen på det skjulte laget og embeddingen i modellen.
- **Bucket**: Totalstørrelsen på tabellen allokert for n-gram tokene.
- **Loss**: Settes til *Negative sampling* (ns), *Hierarchical softmax* (hs) eller *Softmax*. Vi velger Softmax for modellene. Softmax lager en vektor hvor summen er 1 og hvert element forteller sannsynligheten for hver kategori. Hs er en funksjon som gjør modellen mer effektiv for data med flere kategorier.
- **Ws**: *Window Size*. Vektene i modellen oppdaterer seg i kontekst med en tilfeldig størrelse på mellom 1 og ws.
- **Minn**: Minste lengden på character n-grams.
- **Maxn**: Største lengden på character n-grams.
- **Verbose**: Hvordan progresjonen skal visualiseres i terminalvinduet.

Minn og maxn er N-grams på bokstavnivå (eng: *charater N-grams*) og er en av karakteristikkene for fastText. Det er nyttig for å koble sammen ord som er like, men som for eksempel har forskjellige endelser. Det er også nyttig for å beholde sekvens i BoW modellen. [45]

P@1 og R@1

FastText returnerer to resultater etter testing med testsettet; *Precision at one* (P@1) og *Recall at one* (R@1). *Precision* og *Recall* ble forklart i teoridel 2.2.17 som alternative beregninger til *Accuracy* (Nøyaktighet).

P@1 returnerer antall riktige labeler blant alle labelene som ble estimert av fastText. R@1 returnerer antall riktige labeler estimert blant totalt antall riktige labeler. Når hvert eksempel kun får èn label, blir R@1 lik P@1 ettersom antall gjettet av fastText er 1 og antall totalt riktige er 1 per klasse. Når hvert eksempel kun får èn label blir også *Accuracy* (nøyaktighet) det samme som P@1 og R@1. Resultatet P@1 vil derfor brukes i denne rapporten som nøyaktighet.

FastTexts pythonbibliotek

FastText tilbyr et pythonbibliotek for å aksessere funksjoner for å trene modell. En av funksjonene vi bruker er *train_supervised* som trener en modell i form av veiledet læring. Det eneste som kreves av input til funksjonen er treningssettet klargjort for fastText. Vi har også mulighet til å sende inn hyperparameterne dersom vi ønsker å bruke andre verdier enn standard. Hvis vi ikke sender inn hyperparametre, bruker fastText verdiene i tabell 1 (lr står for Learning rate).

Epoch	lr	wordNgrams	minCount	dim	ws	minn	maxn	loss	bucket
5	0.1	1	1	100	5	0	0	softmax	2000000

Tabell 1: FastTexts Standard Hyperparametre

Valg av hyperparametre avhenger av hvilket datasett som benyttes. Vi brukte først standardverdiene i fastText og så hvilken nøyaktighet dette ga på testsettet. Deretter prøvde vi oss frem for å finne verdier som passet datasettet vi valgte å bruke.

Trene modellen med Amazon treningssettet

Vi brukte fastTexts *train_supervised* for å trene på 3,6 millioner Amazon anmeldelser. Vi brukte hyperparametrne i tabell 2.

Epoch	Learning rate	wordNgrams	minCount	dim	ws	minn	maxn
4	0.25	5	5	5	6	4	7

Tabell 2: FastText hyperparametre med Amazon datasett

I datasettet er det kun to kategorier: positiv og negativ. Vi testet med testsettet vårt som består av 30 000 eksempler.

Trene modellen med IMDb anmeldelser

IMDb (Internet Movie Database) datasettet er hentet fra Kaggle [46] som en del av Googles *Word2Vec Sentiment Analysis tutorial*. Den består av 50 000 anmeldelser som i utgangspunktet har en rating på 1 til 10, hvor 10 er best. Datasettet har blitt prosessert slik at alle anmeldelser mindre enn 5 er rangert negativ og over 6 er rangert positiv. Det inkluderer to kategorier som er jevnt fordelt og inneholder 25 000 anmeldelser hver.

Amazon datasettet er på riktig format for å kunne brukes i fastText, men IMDb består av linjer hvor første tegn er 1 eller 0 basert på om det er positivt eller negativt. Teksten i IMDb er heller ikke preprosessert, så først fjerner vi alle linker og tegnsettinger. Deretter fjerner vi mellomrom på starten og slutten av hver anmeldelse. Til slutt legger vi ratingen (kategorien) etter en "__label__" tag etterfulgt

av anmeldelsen, lagret på en tekstfil slik at vi kan bruke den til å trene modellen i fastText.

Eksempel på positivt anmeldelse etter preprosessering: *"__label__2 Although I generally do not like remakes believing that remakes are waste of time this film is an exception [...] I was certain that they are good actors [...] I recommend this film to be on your top 50 films to see and keep on your DVD shelves"*.

Eksempel på negativt anmeldelse etter preprosessering: *"__label__1 I dont know what would be so great about this movie Even worse why should anyone bother seeing this one First of all [...] strong feeling laughter cry fear but in my opinion [...] it lacks all the above But is this worth [...] For me the answer is no"*.

Vi trente fastText på 50 000 IMDB anmeldelser med hyperparameterne i tabell 3 og testet med de 30 000 anmeldelsene fra Amazon testsettet.

Epoch	Learning rate	wordNgrams	minCount	dim	ws	minn	maxn
15	0.25	5	5	100	5	4	7

Tabell 3: FastText hyperparametre med IMDb datasett

3.2 Norske datasett

Vi ønsket å finne datasettet mest egnet til norsk sentimentanalyse. Derfor testet vi to datasett hver for seg, for så å kombinere de. Datasettene har tre verdier: Positiv, negativ og nøytral. Vi har også undersøkt om stemming av ord gir bedre resultat.

3.2.1 Trene fastText på datasettet NoReC

The Norwegian Review Corpus (NoReC) er laget for trening av modeller for sentimentanalyse på dokumentnivå. 35 200 anmeldelser i fulltekst er hentet fra de største norske nyhetsbyråene og er vurdert med terningkast. NoReC ble laget som en del av *SANT (Sentiment Analysis for Norwegian Text) initiative* for Universitetet i Oslo. Det er et prosjekt som ønsker å tilby ressurser og verktøy for sentimentanalyse på norsk språk. Datasettet ble laget i samarbeid med NRK, Schibsted og Aller Media som tilsammen gir oss de største nyhetsbyråene i Norge. [47]

Eksempel på terningkast 5:
"*Led stemningslys kan også gjøres enkelt og rimelig det finnes . . . [527 ord]. . . og prisen for pæra og fjernkontrollen er 249 kroner hos clas ohlson og du trenger ikke koble noen boks til ruteren eller installere noen som helst app. Noen ganger er det enkle en helt utmerket løsning. Vi har også testet en enklere løsning fra philips bestående av lyspære og fjernkontroll. Les vår test av philips living whites.*" *

NoReC tilbyr skripter i Python for å aksessere HTML filene som representerer dataene. Vi kopierte main.py og misc.py inn i prosjektet vårt for å bruke funksjonene i eget skript. NoReC tilbyr også muligheten til å importere norec som en modul [48]. For å aksessere datasettet brukte vi *load* og *html_to_text* fra main.py, slik som i kodesnutt 3.3. Verdien *subset* spesifiserer om vi ønsker å hente ut datasettet til trening, validering eller testing.

Listing 3.3: Aksessere NoReC datasettet med

```
t_data = load("html.tar.gz", subset=subset)
# Convert html to text and put in a list with rating
train_set = [(html_to_text(html), metadata['rating'])
    for html, metadata in t_data]
```

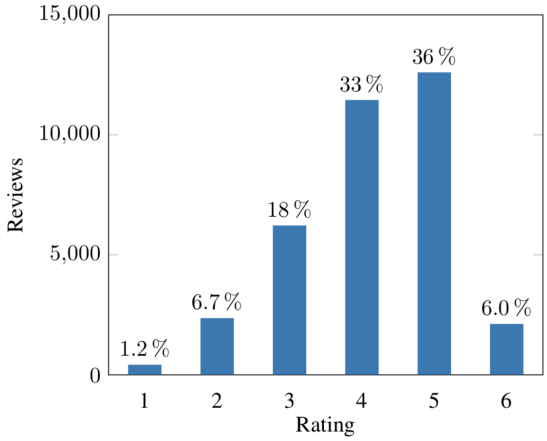
Sentiment	Antall eksempler
Positiv	14725
Nøytral	17677
Negativ	2787

Tabell 4: Fordeling av sentiment i datasettet NoReC

*Eksempelet er kuttet ned med 527 ord.

Terningkast

Allerede i 1952 ble det introdusert terningkast i en filmanmeldelse for VG og dette har blitt standard for norske anmeldelser [47]. Vi valgte å si at 1-2 er negativ, 3-4 er nøytral og 5-6 er positiv. Denne forutsetningen har vi besluttet etter å ha lest artikkelen om hvordan P3s filmpoliti gir terningkast. P3s filmpoliti gir en god representasjon for NoReC datasettet ettersom de anmelder film, serier og bøker. P3s anmeldelser er også inkludert i datasettet. Det ble forklart i artikkelen at terningkastene stod for følgende: katastrofalt (1), dårlig (2), middelmådig (3), god (4), meget god (5) og "en klassiker" (6) [49]. Det Norske Akademis Ordbok beskriver middelmådig som *"middels eller (især) under middels med hensyn til evner, dyktighet, utførelse, kvalitet e.l."* [50].



Figur 5: Fordeling av anmeldelser i NoReC datasettet. Figuren er hentet fra NoReCs forskningsrapport.

Trening

Vi velger å bruke fastText til å trene en modell på NoReC datasettet. Kategoriene er satt til 0 for negativ, 1 for nøytral og 2 for positiv. Vi kunne også valgt å bruke de tekstlige kategoriene, men det ville økt størrelsen på datasettet. Vi deler datasettet i to: Et treningssett med 28158 eksempler (80%) og et testsett med 7031 eksempler (20%). NoReC er allerede delt inn i tre *subsets*: Trening, test og *development*. FastText bruker ikke *development* så vi flytter eksemplene til testsettet. Vi bruker hyperparameterne i tabell 5.

Epoch	Learning rate	wordNgrams	minCount	dim	ws	minn	maxn
25	0.25	3	2	10	5	5	7

Tabell 5: FastText hyperparametre med NoReC datasett

3.2.2 Lage eget norsk datasett

I de neste delkapitlene forklarer vi hvordan vi produserte et egendefinert norsk datasett annotert for sentimentanalyse ved hjelp av Twitter API.

Twitter API

Vi valgte å benytte oss av tweets for å lage et eget datasett, som ga oss enkel og gratis tilgang til mye norsk data. I APIet til Twitter er det ikke er mulig å hente data lenger enn 7 dager bakover i tid. Kontinuerlig uthenting ga oss et grunnlag på litt under 600 000 tweets som ble brukt i produksjonen av datasettet. En tweet besto av maksimalt 140 tegn, men i nyere tid har dette blitt utvidet som gir oss enkelte ufullstendige setninger ettersom tweeten ender i "...".

Hvordan utføre stemming

Primært undersøkte vi tre ulike fremgangsmåter for å stemme (def.: Finne stammen av ordet og så fjerne endelsen, fra teoridel 2.2.10) ord: N-Gram stemmer, ordlistebasert stemmer og Porter stemmer.

N-Gram stemming er en algoritme som ikke tar høyde for språk i utgangspunktet, men som må analysere eksisterende tekst for å lage en indeks over alle N-Gram i teksten. Dette gjorde stemmeren veldig kompleks, samt at den krevde store mengder med minne. Problemet med ordlistebasert stemmer er at man må finne en ordliste og stemmingen vil ta mer tid. For hvert ord som skal stemmes må algoritmen i verste fall lete gjennom hele listen før den lykkes med å finne en ordstamme. Når man skal analysere alle ord i datasettet vil dette være tidkrevnde. [5]

Vi endte derfor opp med den heuristiske algoritmen Porter som baserer seg på gitte regler for å fjerne endelser på ordene. Det er utviklet flere Porterstemmere, men den mest populære algoritmen er Snowball stemmer. Implementasjonen av starter ved å finne området etter første konsonanten etterfulgt av en vokal, eller et tomt område på slutten av ordet om dette ikke finnes (kalt R1). Videre så ser vi etter en rekke endelser i R1 opp mot en liste (heter, ande, er osv.). Om dette ikke finnes, ser vi etter en s-ending i R1 som defineres som en rekke bokstaver (b, c, g, h osv.) eller k om den ikke etterfølger en vokal. Om dette ikke er med i R1 så ser vi om det er endelsen (erte, ert) hvorav vi bytter dette ut med er. I neste steg finner vi (vt, dt) og fjerner eventuelt t-en fra denne endelsen. Siste steg ser etter noen få ekstra endelser som i tilfeller kan ha endelser fra de forrige stegene før seg, og sletter disse fra R1. Alt av endelser ser vi kun etter i enden av R1, og når vi er ferdige plasseres den nye R1 tilbake på enden av ordet, og ordet ses på som stemt. [51]

Preprosessering

For å analysere dataene trenger skriptet vårt en inputfil og et filnavn som resultatene kan skrives til. Det kan spesifiseres om en vil ha enten stemt eller ustemt output. I utviklingen av skriptet implementerte vi både under- og oversampling slik at vi i fremtiden kunne benytte oss av begge vektingsformene i datasettet.

Vi starter med filtrering av tweets, fremover kalt dataene, for å forsikre at tweeten inneholder norske ord. Vi begynner prosessen med å lage en stemt versjon av dataene hvor vi sjekker denne opp mot stemte, norske ordlister for å filtrere bort de som ikke er norske eller som kun består av tegn o.l. Vi har brukt GNU sin Aspell ordliste, online.no sin banneordliste, SNL sin liste over fellesnavn [52] og AFINN ordliste for sentiment [53].

Vi valgte å stemme ordlistene av den grunn at analysen med ustemte ordlister var tidkrevende. Hver linje ble i verste fall sjekket mot nesten 580 000 ord fra ordlisten. Den ferdigstemte ordlisten hadde en størrelse på 110 000 ord. For å forbedre ytelsen ytterligere opprettet vi en liste over de vanligste ordene slik at ved neste kjøring ble ordlisten, som hovedparten av ordene ble hentet fra, redusert til

litt under 2000 ord. Etter dataene er filtrert fjernes uønskede tegn, URLer, hashtagger og alfakrøller for å få de redusert til normal tekst.

Svak, automatisk annotering

Vi annoterer tweetene ved å hente ut ord og emojier som har sentiment og summerer så disse for å få et totalsentiment for hele tweeten. Vi bruker leksikon for sentiment og et bibliotek for emoji-sentiment. Emoji-biblioteket [54] består av emojier som er gitt positiv eller negativ verdi. Vi bruker et leksikon av AFINN [53] sammen med et leksikon av Yanqing Chen & Steven Skiena [55] for å finne sentimentet på ordene i tweeten. Summen av sentiment hos ordene i tweeten og sentiment gitt av emojiene gir scoren for tweeten. Dersom scoren er over 0.15 blir tweeten annotert som positiv og om den er under -0.15 blir den annotert som negativt. Er scoren mellom -0.15 og 0.15 blir tweeten annotert som nøytral.

Lagring

En av de vanligste lagringsformene for datasett i maskinlæring er i en csv-fil. Csv står for *Comma Separated Values* og kan inneholde en rekke verdier. Filen representerer data som kan settes opp i kolonner/-celler, hvor øverste linje er navnene til hver kolonne. Vi valgte lagringsformen fordi det finnes enkle metoder for å gjøre om csv-filer til *dataframes* som enkelt kan benyttes i maskinlæringskoden [56]. Alternativet er å bruke en tekstfil og manuelt opprette dataframes i koden.

3.2.3 Trene fastText på eget datasett

Trener fastText med det ustemte datasettet, med hyperparameterene gitt i tabell 6. Vi trener fastText med det stemte datasettet, med hyperparameterne i tabell 7.

Epoch	Learning rate	wordNgrams	minCount	dim	ws	minn	maxn
25	0.05	3	3	100	128	3	7

Tabell 6: FastText hyperparametre med eget, ustemt datasett

Epoch	Learning rate	wordNgrams	minCount	dim	ws	minn	maxn
25	0.05	3	2	100	5	2	7

Tabell 7: FastText hyperparametre med eget, stemt datasett

3.2.4 Trene fastText på mikset datasett

For å forsøke å få god kvalitet på modellen sjekker vi hvilken nøyaktighet vi får ved å kombinere vårt eget datasett (ustemt) med NoReC. Under kombineringen mikses datasettene tilfeldig slik at NoReC anmeldelsene og tweets blir jevnt fordelt. Vi trener fastText med hyperparameterne i tabell 8.

Epoch	Learning rate	wordNgrams	minCount	dim	ws	minn	maxn
5	0.25	3	3	100	128	3	7

Tabell 8: FastText hyperparametre med mikset datasett

3.3 Modeller for norsk sentimentanalyse

I tillegg til å bruke IBM Watson NLU og fastText på ulike datasett utforsker vi alternativer for modeller på norsk. Dette krever derimot mer kunnskap innen maskinlæring å sette opp.

3.3.1 Googles kurs i maskinlæring

For å lære om maskinlæring, fant vi en informativ opplæring på Google Developers som hadde fri gjenbruk beskrevet under Creative Commons 3.0-lisens for tillatelse. Den skilte seg ut i forhold til andre opplæringsmetoder og ga oss forståelse innenfor maskinlæring og tips til modelleringen. [7]

3.3.2 Programmeringsspråk: Python

Python er en av de mest kjente programmeringsspråkene brukt til maskinlæring. Det er et språk med utfylte og kvalitetssikrede biblioteker for maskinlæring. Samtidig er det enkelt for matematikere, fysikere og professorer i statistikk som ikke har mye kjennskap til programmering å ta i bruk.

Vi bruker bibliotekene Numpy, Pandas og Scikit-learn i Python. Numpy brukes til mer avanserte matematiske funksjoner som ikke er standard i Python. Pandas brukes til å hente ut store mengder data fra filer. Den tilbyr funksjoner for å hente ut kolonner og rader i en csv-fil, og overføre det til en DataFrame variabel i Python. Gjør det mulig å utføre operasjoner på datasettet, som *tokenization* og *embedding* [57]. Scikit-learn er et eget bibliotek for maskinlæring i Python. Vi bruker funksjonen "train_test_split" for å splitte datasettet vårt og få det på riktig format [58].

3.3.3 Tensorflow med Keras

Vi brukte Tensorflow med Keras for å bygge egen modell. Tensorflow er en av de mest brukte maskinlæringsbibliotekene i verden, og er veldig fleksibel. I vårt tilfelle er den for fleksibel i forhold til oppgavens begrensning. Derfor bruker vi Keras som et høynivå grensesnitt til Tensorflow. [59]

Tensorboard kjører i nettleseren på brukerens lokale maskin og tilbyr en rekke verktøy for å forstå, debugge og optimalisere Tensorflow modellen. Når modellen har trent lagrer vi informasjon av treningen i en loggfil [60]. Vi brukte Tensorboard til å forstå oppbyggingen av modellen vår og for å se treningsutviklingen representert som en graf.

3.3.4 Jupyter Notebook

Jupyter Notebook er en *open-source* webapplikasjon for å lage dokumenter med live kode, visualiseringer og likninger sammen med ordinær tekst. Den lar deg kjøre deler av en kode flere ganger uten å kjøre hele filen. Dette er praktisk når man skal laste inn store datasett i starten av modellen. [61]

3.3.5 Maskinvare

Macbook Pro

For å kjøre fastText modellene brukte vi personlig PC ettersom det ikke var veldig tidkrevende. Det er en Macbook Pro med 2 kjerner, 3.6 GHz prosessor (Intel Core i5) og 8 GB 2133 MHz minne. Operativsystemet er macOS Mojave versjon 10.14.3.

Virtuell maskin

For å kjøre LSTM modellen trengte vi en kraftigere maskin for å effektivisere kjøringen. Vi kjørte så treningen av modell på en virtuell maskin på NTNU IDI basert på Linux. Den har 32 kjerner, 2,1 GHz prosessor (Intel Xeon E5-2620), 240 GB minne og operativsystemet er Ubuntu 18.04.2 LTS.

3.3.6 Oppbygging

For å lage en modell må en laste opp data og splitte opp i et treningsett og et testsett. Vi bruker Scikit-learns *train_test_split* som gir oss en liste over treningsdata, treningskategorier, testdata og testkategorier. Vi velger en *random_state* og beholdes som en fast størrelse gjennom utforskningen. Vi henter ut eksempler fra datasettet tilfeldig og fordeler det til test eller treningsettet. Ved å bruke en tilfeldig tilstand (random_state) velger vi de samme eksemplene til testing og trening for hver kjøring. Kodesnutt 3.4 viser hvordan vi gjennomfører dette. Vi har tatt utgangspunkt i koden i en opplæring for Tensorflow Keras modellering av Magnus Erik Hvass Pedersen. [62]

Listing 3.4: Laste opp data og splitte i sett.

```
# Load data
data = pd.read_csv('data.csv', sep=';')
X = data["text"]
y = data["sentiment"]
x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=123)
```

Det neste er Tokenizing (def.: Gjøre om fra tekst til tall, fra teoridel 2.2.11). Vi bruker Keras sin Tokenizer i kodesnutt 3.5.

Listing 3.5: Tokenizing.

```
# Tokenizer
num_words = 10000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
x_train_tokens = tokenizer.texts_to_sequences(x_train)
x_test_tokens = tokenizer.texts_to_sequences(x_test)
```

Neste steg er å gjøre alle tokener like lange ved å bruke *padding* og *truncating*. Se kodesnutt 3.6. Vi velger at padding/truncating skal forekomme foran i tokenet, ved å sette pad='pre'. Valgt lengde for tokener finnes ved å ta gjennomsnittlig lengde av alle tokener pluss to standardavvik.
Vi velger å bruke Keras *Sequential model*, hvor en legger til lagene sekvensielt. Så legger vi til embedding i modellen og velger en størrelse for embeddingvektoren.

Så velges hvilke lag modellen skal bestå av. LSTM (*Long-Short Term Memory*) er en type node for RNN-modeller som er brukt i sentimentanalyse [63]. Tensorflows Keras tilbyr LSTM lag, med muligheter til å velge antall nevroner og verdi for dropout.

Datasett

Vi bruker det ustemte datasettet av tweets for trening av modell i LSTM (3.2.4). Datasettet er lagret i en CSV-fil før det blir lastet opp i koden. Tweetene er annotert med 0 (negativ), 1 (nøytral) eller 2 (positiv).

Listing 3.6: Tokenizing.

```
# Padding
num_tokens = [len(tokens) for tokens in
x_train_tokens + x_test_tokens]
num_tokens = np.array(num_tokens)
max_tokens = np.mean(num_tokens) + 2 * np.std(num_tokens)
max_tokens = int(max_tokens)

pad = 'pre'
x_train_pad = pad_sequences(x_train_tokens,
maxlen=max_tokens, padding=pad, truncating=pad)
x_test_pad = pad_sequences(x_test_tokens,
maxlen=max_tokens, padding=pad, truncating=pad)
```

Ett lag LSTM

Koden i 3.7 viser hvordan LSTM lag kan bli lagt til en modell før trening. Laget *Dense* bruker vi fordi vi ønsker tre diskrete verdier: Positiv, nøytral og negativ. Vi vil vite sannsynlighetsfordelingen til hvert eksempel, og derfor velges aktiveringsfunksjonen softmax.

Listing 3.7: LSTM lag

```
# Layers
model.add(LSTM(units=64, dropout=0.6, recurrent_dropout=0.4))
model.add(Dense(3, activation='softmax'))
```

Koden i 3.8 forklarer at vi bruker optimizer Adam og lossfunksjon Sparse Categorical Cross Entropy, som brukes til klassifiseringsmodeller med flere kategorier. Vi bruker i denne modellen 5 epocher og en læringsrate på 0,001 (som er standard i optimaliseringsalgoritmen Adam).

Listing 3.8: Kompilere, trene og evaluere modell

```
optimizer = Adam()
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])
model.fit(x_train_pad, y_train, validation_split=0.05, epochs=6,
batch_size=512, callbacks=[tensorboard])
result = model.evaluate(x_test_pad, y_test, batch_size=512)
```

3.4 Systemutvikling

3.4.1 GitHub

Ettersom det var tre fra gruppen som skulle jobbe med å utvikle koden samtidig og vi ønsket å ha koden lagret eksternt, var det naturlig å benytte seg av et versjonskontrollsystem. Det sto mellom to ulike teknologier, GitLab og GitHub. Gruppen benytter seg regelmessig av GitHub, og framfor å bruke tid på å sette seg inn i en ny teknologi gikk vi for denne. GitHub er en *open source repository hosting* service, som *hoster* kildekoden til prosjekter og gjør det mulig for utviklere å gå tilbake i koden og se på endringer gjort, se 2.4.3. [64]

3.4.2 Travis CI

I forbindelse med at gruppen valgte GitHub som versjonskontroll system, ønsket vi et eksternt verktøy for kontinuelig integrasjon ettersom GitHub ikke har dette integrert. Flere ulike teknologier blir foreslått av GitHub for dette, som automatisk synkroniserer seg med versjonskontrollsystemet. Gruppen har gjennom studieløpet jobbet med Travis CI, så vi undersøkte mulighetene for å benytte oss av denne. Travis CI er en kontinuerlig integreringstjeneste som brukes til å bygge og teste prosjekter som hostes av GitHub [65]. Den bygger automatisk koden når den overføres til Github og varsler dersom det er feil i koden slik at utvikler varsles og forhåpentligvis retter opp i feil før det blir sendt ut i produksjon [66].

3.4.3 IBM Cloud

Ettersom IBM er oppdragsgiveren vår, ønsket vi å teste ut og prøve ressurser vi fikk fra de, der i blant IBM Cloud og ulike skytjenester den hadde å tilby.

Natural Language Understanding

Som nevnt over ønsket gruppen å benytte seg av IBM Cloud sine skytjenester i utviklingen av dette systemet, og trengte i forbindelse med dette en teknologi for å analysere nyhetsartiklene. Valget falt da på IBM Watson sin *Natural Language Understanding*, NLU. NLU tilbyr et sett med tekstanalysefunksjoner som kan brukes til å analysere tekst, der i blant følelser og sentiment analyse [67].

Cloudant

Ved valg av database, var det også naturlig for oss å velg en IBM ressurs. Her var det flere muligheter, men det var primært tre vi vurderte: Databases for MongoDB, DB2 og IBM Cloudant. Databases for MongoDB ble fort utelukket, da denne ikke var gratis. Både DB2 og IBM Cloudant tilbød et vist antall MB uten kostnad, samt et vist antall koblinger. Valget falt til slutt på IBM Cloudant da tjenesten tilbød 1 GB datalagring [68], i motsetning til DB2 som kun tilbød 200 MB [69]. IBM Cloudant er en JSON, ikke-relasjonell, distribuert database, så gruppen var kjent med oppsettet til databasen og hvordan den skulle brukes.

Cloud Foundry

For å distribuere og kjøre applikasjonen vår valgte vi å bruke IBM Cloud Foundry. Den tilbyr kontinuerlig utrulling med god oversikt over applikasjons utrullings stadiene: bygging, staging og produksjon [70]. Dette gjorde det enkelt for gruppen å sette applikasjonen ut i produksjon, kjøre gjennom testene og få gjort den tilgjengelig på internett regelmessig.

3.4.4 React

Ettersom gruppen ønsket å utvikle en applikasjon ved hjelp av Javascript var det spesielt to ulike teknologier vi ønsket å velge mellom, React og Angular. Gruppen har i løpet av vår semesteret 2018 hatt litt om og praktisert React i faget TDAT2004 - Datakommunikasjon med nettverksprogrammering. Vi følte oss derfor tryggere på denne teknologien og ønsket heller å sette oss inn i maskinlæring og trening av egen modell enn et nytt rammeverk. Dette var en av grunnene til at valget falt på React. Gruppen ønsket i tillegg å bruke et rammeverk som er populært i dagens utviklermiljø, samt relevant for fremtidig jobb. React er et moderne, relativt nytt og stadig voksende *frontend* rammeverk. Det er populært og blir for tiden mye brukt innefor frontend utvikling. En annen grunn til at valgte falt på å bruke React er måten man utvikler på. Man bygger små, gjenbrukbare komponenter med en deklarativ tilnærming, der man uttrykker hvordan applikasjonen skal se ut fremfor å angi hvilke DOM-operasjoner som er nødvendig for å oppnå en man ønsker [71]. React sin konseptuelle modell er også meget enkel å forstå, som gjør det enklere å sette seg inn i teknologien ettersom vi ikke hadde så stor erfaring med den fra før [71].

3.4.5 Context API

Context API er React sin innebygde metode for å håndtere state på tvers av komponenter. Den tilbyr en måte å sende data gjennom flere lag med komponenter uten å måtte sende med props manuelt på hvert eneste lag [72]. Vi så også på, og vurderte å bruke Redux, men ettersom dette er mer egnet for komplekse systemer så vi ikke behovet for dette i vårt mindre komplekse system.

3.4.6 Node.js

Node.js er et JavaScript runtime-system, designet for å bygge nettverksapplikasjoner [73]. Det bruker JavaScript på serveren, som er en av grunnene til at vi valgte å bruke denne teknologien. Vi ønsket å ha et felles språk for frontend og backend, som vi fikk til ved å bruke Node.js på backend og ReactJS på frontend. I tidligere prosjekter har gruppen også benyttet seg av Node.js, som er den andre grunnen til at vi valgte å bruke det, ettersom vi er familiere med det.

3.4.7 JEST

Valget på hvilke teknologi vi skulle bruke til testing sto primært mellom to stykker, JEST og Mocha. Gruppen ønsket noe som var lettvindt, familiert og støtter prosjekter som er bygd på React og Node. Mocha ble først vurdert, ettersom dette er noe vi har jobbet med før. Men den ble valgt bort ettersom den krever andre rammeverk, som chai, for å kunne teste systemet komplett. Valget falt da på JEST, et JavaScript test rammeverk som fungerer på prosjekter som blant annet Node og React [74]. I tillegg er det en pakke som inneholder alle deler en trenger til testing, man trenger altså ikke inkludere andre rammeverk, slik som en trenger med Mocha.

3.4.8 Babel

For å ha muligheten til å kjøre systemet vårt i en hvilken som helst nettleser, valgte vi å bruke Babel for å konvertere *edge JavaScript* til den eldre versjonen *ES5 JavaScript* som støttes av de fleste nettlesere. Babel støtter de nyeste JavaScript versjonene gjennom syntaks reformer og lar oss bruke ny syntaks uten å måtte vente på at nettleserene skal støtte den [75].

3.4.9 NewsAPI

Applikasjonen er avhengig av å ha tilgang til nyhetsartikler, helst fra hele verden. Etter å ha lest oss opp på ulike verktøy som kan gi oss tilgang på dette, kom vi over *News API*. Systemet får da enkelt tilgang til omfattende og oppdatert nyhetsdekning fra kilder over hele verden [76].

3.4.10 Auth0

Under utviklingen av applikasjonen bestemte gruppen seg for å legge til muligheter for en adminside. Dette fordi vi ønsket at det skulle være mulig for en bruker å legge til nyhetskilder de ønsker skjermskrapet. For at ikke alle brukere skulle ha tilgang til skjermskraperens kontrollenhet ønsket vi å bruke en teknologi som kun ga visse epost adresser tilgang. Valget sto her mellom to ulike teknologier, Auth0 og IBM sin Identitets- og tilgangshåndtering, IAM. IAM hadde flere begrensninger, som et vist antall innlogginger i måneden, som kunne blitt problemtaisk under utviklingen. Valget falt derfor på Auth0. Auth0 tilbyr autentisering og autorisasjon som en tjeneste og kan kobles til et hvilket som helst program [77], og begrensningen de har på antall innlogger er såpass stor at den ikke vil påvirke utviklingen eller testingen av systemet.

3.4.11 Trello

Bachelorgruppen bestemte seg for å benytte seg av utviklingsmetodikken Scrumban, som blant annet krever at oppgaver og mål blir ført opp på en tavle med følgende kategorier: *Backlog: To do*, *In progress*, *Done*. Vi så først på å lage en slik tavle ved hjelp av postit lapper, men innså at dette kunne ende med mange lapper og rot. Derfor bestemte vi oss for å se etter en virtuell løsning. I et møte med teknisk veileder fra IBM, Bjørnar Kjenaas Mælum, fikk vi anbefalt Trello som de pleide å bruke under utvikling av systemer. Trello er et sammarbeidsverktøy der medlemmer kan lage lister, kategorier og kort med oppgaver som skal gjøres [78], så det er enkelt for et team og få oversikt over hva som jobbes med og hva som må gjøres. Gruppen delte inn tavlen i fem kategorier som vi mente var relevant for vårt prosjekt: *Backlog: To do*, *To do*, *In progress*, *Done* og *Bugs*.

3.4.12 Designteknikker

Under utførelsen av dette bachelorprosjektet var det mye fram og tilbake om hvem bruker av systemet skulle være. Men etter å ha diskutert og kommet fram til en endelig problemstilling, ble målgruppen satt til å være forskere som ønsker å studere trender i mediebildet. I forbindelse med dette ønsket vi også å tilpasse brukergrensesnittet og designet til å passe denne typen bruker og fokusere på å vise fram statistikk og data knyttet til sentimentanalyse. Vi undersøkte derfor hvordan en på best mulig vis kan vise fram store mengder data og presentere resultater. Det første vi så på var hvordan best presentere store mengder data, og gjøre dette forståelig for bruker. Dette ettersom resultater av sentimentanaylsene er gjort på flere tusen artikler. Vi kom da over informasjons visualisering, skrevet om i 2.4.8, som omhandler hvordan en på best mulig måte kan presentere data grafisk ved hjelp av grafer, linjediagram og lignende. Vi valgte å benytte oss av denne formen for visualisering i form av *donut grafer*, for å vise gjennomsnittlig sentimenter eller følelser i mediebildet, og linjediagram, for å vise gjennomsnitt av sentimenter eller følelser per dag i et gitt tidsintervall.

Vi ønsket, som problemstilling 2 tar opp 2, å forenkle og tilgjengeliggjøre sentimentanalyse for å gjøre det lettere å studere trender i mediebildet og ville derfor designet et brukergrensesnitt som på best mulig måte lar en gjøre dette. Derfor valgte vi å sette oss inn i visuelt hierarki, se 2.4.9, og hvordan mennesker pleier å lese over og skanne nettsider. Det sto primær her mellom to ulike lesemønstre, F- og

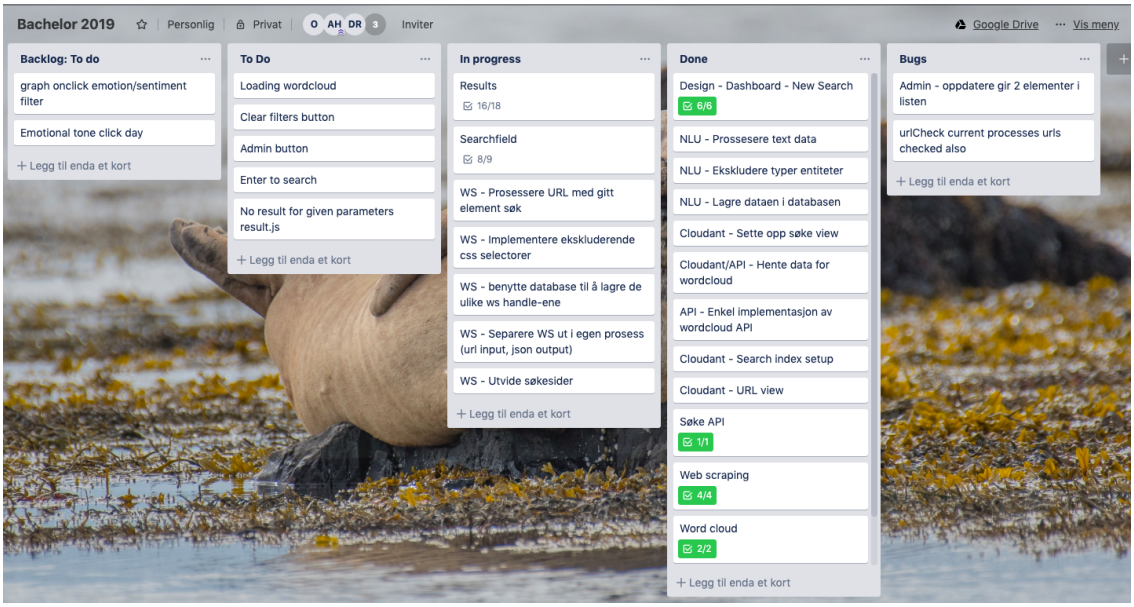
Z-lesemønster, som vi vurderte å tilpasse siden vår til. Ved F-lesemønster skanner bruker siden i form av en F, der vedkommende starter oppe i venstre hjørnet [79]. Men etter å ha lest at dette er noe man ønsker å unngå, ettersom bruker ofte kan overse mye informasjon på nettsiden [79]. Vi valgte derfor konstruere siden med den viktigste informasjonen spredd som en Z, se 3 for illustrasjon.

3.5 Utviklingsprosess

Under utviklingen av systemet ble det benyttet en Agil metodikk kalt ScrumBan, som kombinerer egenskapene fra to velkjente utviklingsprosesser. På den ene siden har vi Scrum, som best egner seg for utvikling og produksjon av et system. Men ettersom dette er en metodikk som er ment for team på 5-8 personer, og generelt er lite hensiktsmessig for små team med alle aktivitetene metodikken består av, ble denne valgt bort. På den andre siden har vi KanBan, en mindre organisert metodikk som brukes til implementering av *Agile* programvareutvikling. Dette var en lite organisert metodikk, uten noe form for sprinter, så valgte vi heller en litt mer organisert utviklingsprosess. Valget falt derfor på ScrumBan.

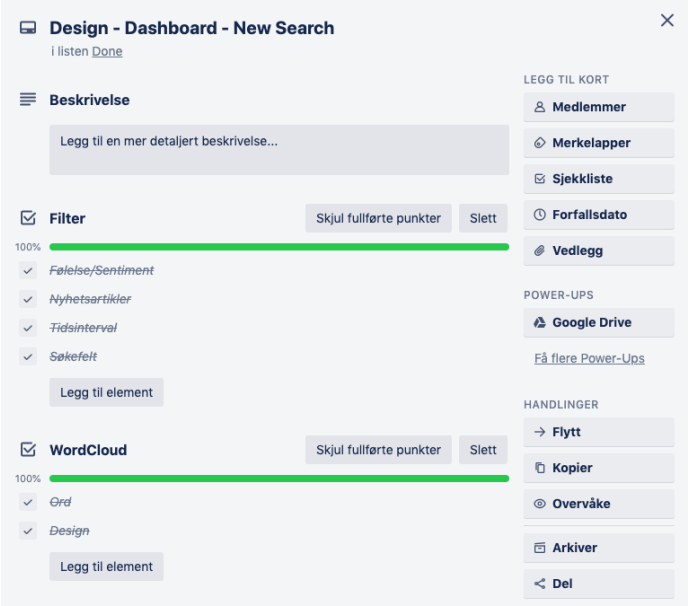
Bacheloroppgaven ble delt inn i Oppstart og 13 Sprinter, alle med forskjellig lengde og fokusområde i Gantt diagrammet (Se prosjekthåndboken [80]). Ikke alle krav til systemet ble satt opp før start, og vi ønsket derfor en metodikk der dette kunne endres underveis. I ScrumBan kan nye oppgaver legges til i en pågående iterasjon og det er valgfritt å sette opp timeestimering. I tillegg er ikke roller definert, men settes basert på gruppens ønsker og antall medlemmer [81]. Dette var viktig for bachelorprosjektet, ettersom gruppen bare bestod av tre medlemmer, hvorav to primært skulle drive med utviklingen.

Vi delte inn i to ulike roller: *Backend* og *Frontend*. Ettersom dette er knyttet til utviklingen av systemet, ble ikke maskinlæring satt opp som en rolle eller brukt under utførelsen av systemutviklingen. Sprinter oppført i Gantt diagrammet ble så slått sammen slik at en sprint inneholdt både *backend* og *frontend*. Videre krever ScrumBan et ScrumBan *Board* der oppgaver skal føres opp (se 3.5) for å gi en oversikt over hvilke oppgaver som har blitt utført og hvilke som må gjøres. Dette var også en av grunnen til at gruppen valgte å benytte ScrumBran framfor KanBan, vi ønsket å ha en oversikt over prosjektet og hva vi hadde igjen av arbeid. Ettersom en fysisk prosjekttavle kunne blitt mye rot, bestemte vi oss heller for en virtuell løsning; Trello. Nedenfor er et skjermbilde av tavlen vår under utviklingen, og hvordan vi har satt opp ulike kort per oppgave.



Figur 6: Skjermbilde av produkttavle for prosjekt satt opp i Trello

En oppgave, et kort, kan bestå av flere underoppgaver knyttet til tittelen på kortet. Dette for å forhindre at en oppgave ikke skulle bli for stor. En oppgave ble satt opp i forbindelse med use case og funksjonelle krav satt i visjonsdokument, som for eksempel Design - Dashbord - New Search. Under et kort er det satt opp flere mindre oppgaver som er knyttet til tittelen på kortet og funksjonalitet nødvendig for å få utført denne. Se figur nedefor, 3.5.



Figur 7: Skjermbilde viser hvordan et kort i Trello kan bestå av flere underoppgaver

3.6 Rollefordeling

Bachelorgruppen har ved flere anledninger jobbet sammen i ulike prosjekter under studiet, og kjenner derfor godt til hverandres arbeidsmetoder, kunnskap og ferdigheter. Vi valgte derfor å gå sammen under bachelorprosjektet ettersom vi har egenskaper som utfyller hverandre. Hver og en av oss har ulike egenskaper og erfaringer innenfor ulike områder i fagfeltet, og fordelte rollene etter dette.

Dina Rosvoll ble tildelt hovedansvaret for maskinlæring og trening av egen modell. Hun begynte studiet uten erfaring med programmering, men har i løpet av studietiden bygd opp en interesse for maskinlæring og ønsker nå å fortsette studieløpet sitt innenfor AI og maskinlæring. Det ble derfor bestemt at hun skulle få hovedansvaret innenfor denne delen, da hun ønsket å lære mer om, og utvikle seg, innen maskinlæring.
Oda Steinland Skaug ble tildelt hovedansvaret for *Frontend Development*. Hun startet på studie uten forkunnskaper for programmering og har under studiet bygd opp en interesse for interaksjonsdesign og *Frontend development*, herav naturlig at hun skulle bli tildelt hovedansvaret for denne delen av prosjektet. Hun tok også ansvaret for dokumentasjon.
Andreas Hammer har hatt hovedansvaret for *Backend Development*. Han er den på gruppen med mest programmeringserfaring, også fra før studietiden, og har mye kunnskap innefor programmering. Med mye erfaring innenfor fagfeltet, samt *Backend development*, og med en evne til å skape funksjonelle, kreative og effektive løsninger var det naturlig at han skulle få hovedansvaret for dette.

4 Resultater

Dette kapittelelt tar for seg resultatene og hva vi har funnet ut under utførelsen av denne bachelor-oppgaven. Første kapittel omhandler resultatene i forskningsdelen. Resultater for systemutviklingen er delt inn i tre: Vitenskapelige resultater, knyttet til produkt og design, Ingeniørfaglig resultater, knyttet til mål satt i visjonsdokument, og administrative resultater, knyttet til framdriftsplan og Gantt diagram.

4.1 Maskinlæring

Først viser vi nøyaktighet på engelske modeller for sentiment analyse og hvordan de analyserer utvalgte eksempler. Deretter kommer et utsnitt av det egenproduserte datasettet og et eksempel på hvordan en tweet ble stemt. Til slutt presenteres nøyaktighetene til de norske modellene i en tabell sammen med treningstid.

4.1.1 Engelsk sentimentanalyse

Modellen trent på Amazon datasettet ga en nøyaktighet på 93,82% etter 716.4 sekunder. FastText modellen trent på IMDb datasettet, men testet på de 30 000 Amazon anmeldelsene ga 82,00% etter 113,1 sekunder. Watson NLU ga 85,68% etter noen timer. En oversikt er oppgitt i tabell 9.

Modell	**Nøyaktighet på samme testsett**	**Tid**
Watson NLU	85,68%	-
FastText Amazon	93.82%	716 sek
FastText IMDb	82.00%	113 sek

Tabell 9: Resultater engelsk sentimentanalyse

Watson NLU returnerer en verdi mellom -1 og 1 ut i fra hvor negativ eller positiv den er. Dersom et estimat er på -0.2 er den 20% negativ. FastText derimot, returnerer resultatet fra softmax som forteller hvor sikker fastText er på estimatet positiv eller negativ.

Eksempel på riktig estimat

- *Poorly Made CD: I liked Danielson's Alpha but unfortunately, Omega doesn't play on my computer. It seems the CD wasn't mastered correctly. Buyer beware![DW]*
    - Watson: 91.6% Negativ
    - FastText (Amazon): Negativ, 100% sikker
    - Riktig: Negativ

Eksempler på feil estimat

- *These don't work off indirect sunlight.: I have no windows with direct sunlight and it will not work. Good Idea but it needs a bigger solar cell.*
  - Watson: 30.6% Positiv
  - FastText (Amazon): Negativ, 99.79% sikker
  - Riktig: Negativ
- *HE slips a little but still a well worth waiting conclusion!: Not as great as every other book but wow what an ending!*
  - Watson: 29.8% Negativ
  - FastText (Amazon): Positiv, 100% sikker
  - Riktig: Positiv
- *I thought is wasn't good.: This is about a boy and a cricket. Mario, the boy, is working at a newsstand when he hears a noise. He looks under a trash can and finds a cricket. He buys a cage for the cricket, and that's how it turns out. The part I liked is when the cricket chirps and asks the man who was going to steal the bell what he was doing. I did not like the rest*
  - Watson: 67.1% Negativ
  - FastText (Amazon): Positiv, 95.4% sikker
  - Riktig: Negativ

4.1.2 Eget norsk datasett

Denne delen forklarer hvordan det datasettet ble. Tabell 13 viser et utsnitt av tweetene etter automatisk annotering. Det er jevn fordeling mellom de tre kategoriene og totalt 172 113 tweets.

Kategori	**Tweet**
Nøytral	&gterrf&gtfye
Nøytral	mamma er håpløs med elektroniske ting jeg har måttet hjelpe henne flere ganger med å justere volum på mobilen henn
Nøytral	pyton skaperen rolf håndstad bedre kjent som rhesus minus er død år gammel
Positiv	denne gjengen har valgt det feteste band navnet vi har hørt på leeenge sjekk ut deres første låt nå på spotify
Positiv	synes du skal holde deg til sport og analyser om det ingenting interessant om støres tale syntes d
Negativ	tror ikke nødvendigvis at det er en dum ting å være opptatt av
Negativ	brutalllll

Tabell 10: Utsnitt av tweetene i datasettet etter automatisk annotering

Stemming

Stemming komprimerte datafilen fra 19,1 MB til 18,1 MB. Eksempelet under viser en tweet før og etter stemming:
Uten stemming: *"pappa var ikke så ufeilbarlig og allvitende likevel gitt helten er egentlig ganske bitter og døll kjipt det gunn"*.
Med stemming: *"papp var ikk så ufeilbar og allvit likevel gitt helt er egent gansk bitt og døll kjipt det gunn"*.

4.1.3 Modeller for norsk sentimentanalyse

Tabell 11 forteller hvilken nøyaktighet vi fikk på de ulike datasettene. De første radene er modeller trent med fastText. Kolonnen binærklassifisering viser klassifiseringen med kun to kategorier, hvor nøytral er fjernet for å kunne sammenligne med de engelske datasettene.

Modell	**Nøyaktighet**	**Tidsbruk**	**Binærklassifisering**
NoReC (Tabell 5)	71,95%	84,8 sek	94,4%
Ustemt (Tabell 6)	81,57%	45,4 sek	95,3%
Stemt (Tabell 7)	79,37%	39,2 sek	94,11%
Mikset (Tabell 8)	77,74%	125,5 sek	93,74 %
LSTM ett lag (kapp. 3.3.6)	78,44%	6 min	*Ikke testet*

Tabell 11: Resultater norsk sentimentanalyse

4.2 Vitenskapelige resultater

I dette delkapittelet vil vi ta for oss de vitenskapelige resultatene. Disse er basert på hva som ble produsert: applikasjonen, designet og valg tatt i forbindelse med dette i lys av **problemstilling 2**; Utvikling av en applikasjon som forenkler og tilgjengeliggjør sentimentanalyse for forskere som ønsker å studere trender i mediebildet.

4.2.1 Produkt

Systemet som ble levert ved bachelorprosjektets ende, 20.05.19, består av en WEB-applikasjon. Applikasjonen blir hostet gjennom IBM Bluemix på platformen Cloud Foundry, og er i dag satt opp og tilgjengelig på følgende link:

http://ado-sentiment-analysis.eu-gb.mybluemix.net/

Miljøvariabler og bygge applikasjon

Applikasjonens kildekode er tilgjengelig, og om man skal ønske å kjøre applikasjonen på egen maskin må man klargjøre den litt i forkant. Dette innebærer å sette opp miljøvariabler og bygge applikasjonen. Av miljøvariabler trenger en å sette opp en SESSION_SECRET, SCRAPER_API_KEY som gir tilgang til å snakke med *webscraperen* uten å autentiserer mot Auth0, ADMIN_WHITELIST som er en kommaseparert streng hvor e-post adressene til alle som skal ha full tilgang til *webscraperen*.

Applikasjonen er satt opp med Auth0 som autentiseringsplatform, og denne må også konfigureres. Dette gjøres ved å sette miljøvariablene AUTH0_DOMAIN, AUTH0_CLIENT_ID, AUTH0_CLIENT_SECRET, AUTH0_CALLBACK_URL som en finner i konfigurasjonen av egen Auth0 løsning.

IBM Cloudant og NLU krever litt annerledes behandling. Her lagrer du Cloudant og NLU sine *credentials* i filen *.ibm-credentials* i rotmappen til prosjektet. Videre så plasserer NewsAPI API-nøkkelen i filen *.ibm-credentials* på liknende form som IBM *credentialsene*, mer om dette i README.md filen til applikasjonen. Når konfigurasjonen er fullført bygges applikasjonen ved å kjøre *npm install && npm run build*. Deretter kjøres den ved å kjøre *npm start*

Funksjonalitet

Systemet som ble leveret 20.05.19 bestod av ulik funksjonalitet knyttet til søk og resultater. Følgende funksjonalitet, knyttet til visjonsdokument og planlagte mål, er tilstedet i applikasjonen ved innlevering:

- **Filtrer før søk**

  En bruker har mulighet til å legge på følgende filter før søk blir utført: Velge mellom sentiment eller tone søk, velge spesifikke nyhetbyråer og velge tidsintervall en ønsker artikler fra.

- **Filtrer etter søk**

  En bruker kan filtrere resultater etter utført søk. En kan da filtrere på følgende: Velge bestemt tidsintervall eller dato en ønsker å se resultatet fra og på ulike følelser eller sentiment. Siste punkt kommmer ann på om man har utført et sentiment søk eller et tone søk.

- **Utføre søk: Sentiment**

  Dersom ikke bruker har valgt noe annet, vil søket utføres som sentiment søk. Dette gjelder også ved klikk på ord i ordsky. Analysen og resultatet vil da være basert på følgende: *Positive, Negative* og *Neutral*.

- **Utføre søk: Tone**

  Bruker kan også, ved å legge på filter, utføre et tone søk. Analysen og resultatet vil da være basert på følgende: *Sadness, Anger, Joy, Disgust* og *Fear*.

- **Ordsky**

  På dashbordet vil det være en ordsky basert på ord og fraser mest nevnt i mediebildet. En bruker kan trykke på ord og få utført et sentiment søk på valgt ord.

- **Navigere til ønsket artikkel**

  Etter utført søk, på resultatside, vil bruker få oversikt over artikler inkludert i søket. En kan da klikke på knapp på artikkel for å bli navigert videre til denne.

- **Se statistikk: Gjennomsnittlig sentimenter eller følelser**

  Basert på om det ble utført sentiment eller tone søk, vil bruker få grafer over gjennomsnittlig følelser eller sentimenter. Disse vil endre seg hvis filter blir lagt på etter søk, og oppdatere seg til å stemme med artikler innenfor tidsintervall valgt.

- **Se statistikk: På hver enkelt artikkel**

  En bruker vil kunne se sentiment eller følelses analyse gjort per artikkel. Hver artikkel vil inneholdet et avsnitt som informerer om dominerende følelse eller sentiment og grafbarer som viser hvor stor prosent andel hver følelse elle sentiment har.

- **Se statistikk: Emosjonell tone innenfor tidsintervall**

  Hver resultatside vil inneholde et linjediagram som viser gjennomsnittet per følelse eller sentiment per dag med i søket som har blitt gjennomført. Hver linje i diagrammet er en følelse eller et sentiment.

- **Printe ut og eksportere resultater til PDF**

  En bruker vil kunne skrive ut og eksportere resultatsiden etter gjennomført søk. Dokumentet vil være som et skjermbilde av skjermen tatt i det bruker klikker på skriv ut knappen.

Tilleggsfunksjonalitet

Under utviklingen av systemet ønsket vi å legge til funksjonalitet tilknyttet administrering, og bestemte oss for å gjennomføre dette dersom det ble tid til overs. Følgende tilleggsfunksjonalitet er tilstedet og mulig å benytte seg av ved innlevering:

- **WebScraper Administrator Tool**

  Dersom bruker har admin tilgang, som vertifiserers gjennom Auth0, har en tilgang på *WebScraper Administrative Tool*. En bruker kan da gjøre følgende:

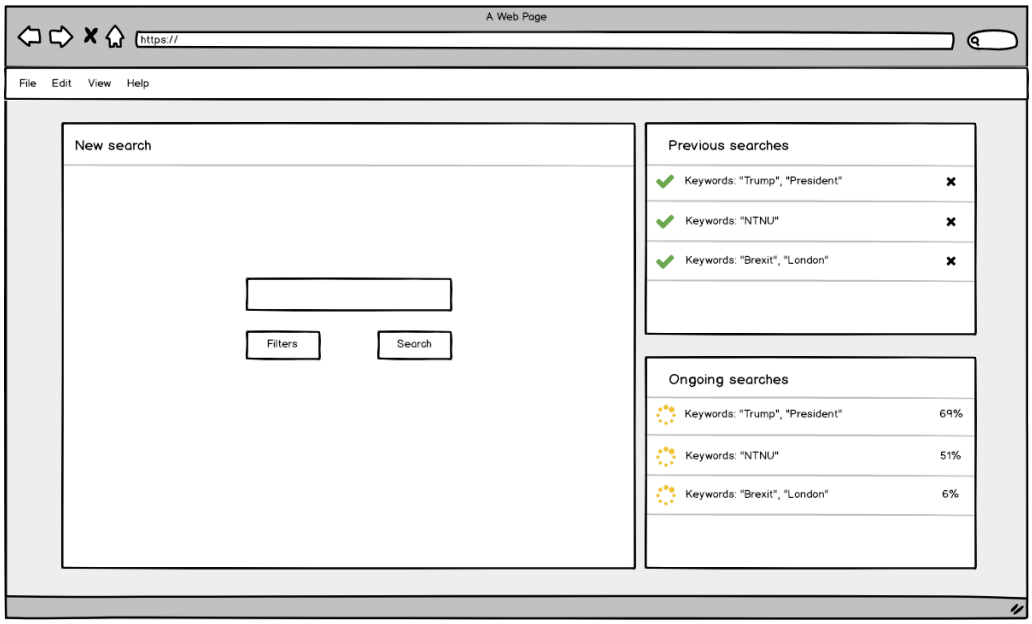
- Endre hosts, legge til hosts og teste hosts

Dersom bruker i tillegg har opphøyde administrative rettigheter, som vertifiseres gjennom miljøvariablene til tjeneren, kan en bruker også gjøre følgende:

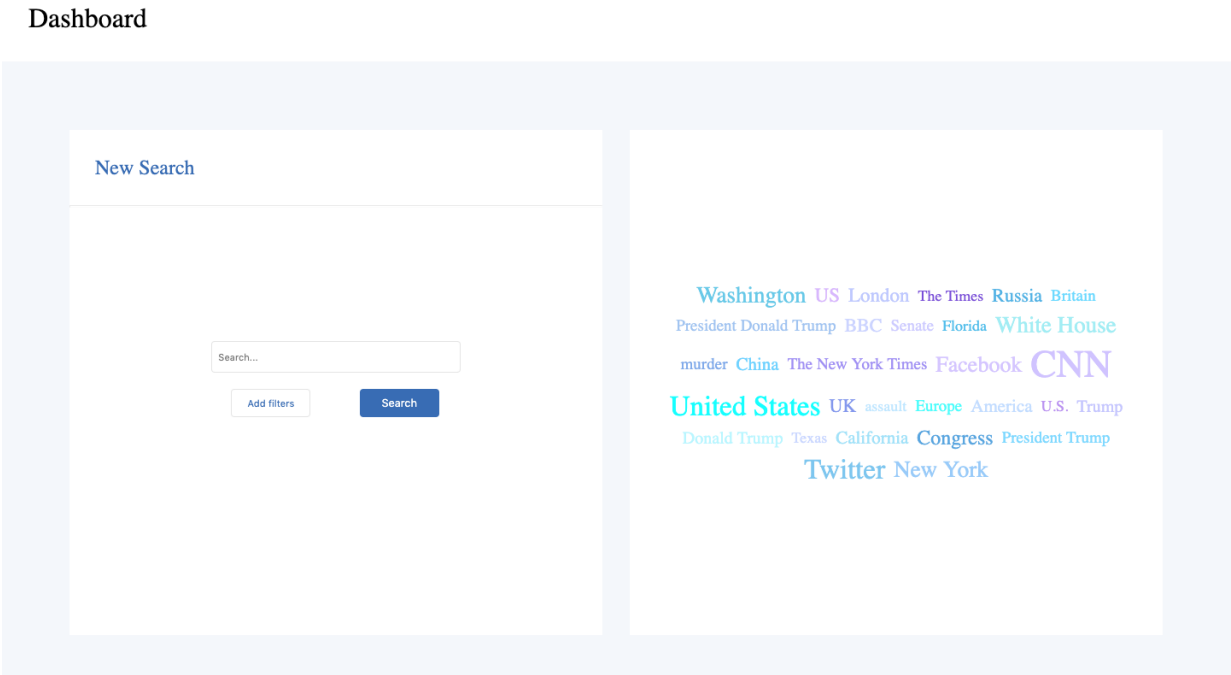
- sette opp tidsbaserte jobber, der webscraper henter nyhetsartikler fra spesifiserte kilder
- Indeksere spesifikke urler
- Hente frem og indeksere de hundre nyeste nyhetsartiklene fra spesifikke nyhetskilder.

4.2.2 Design

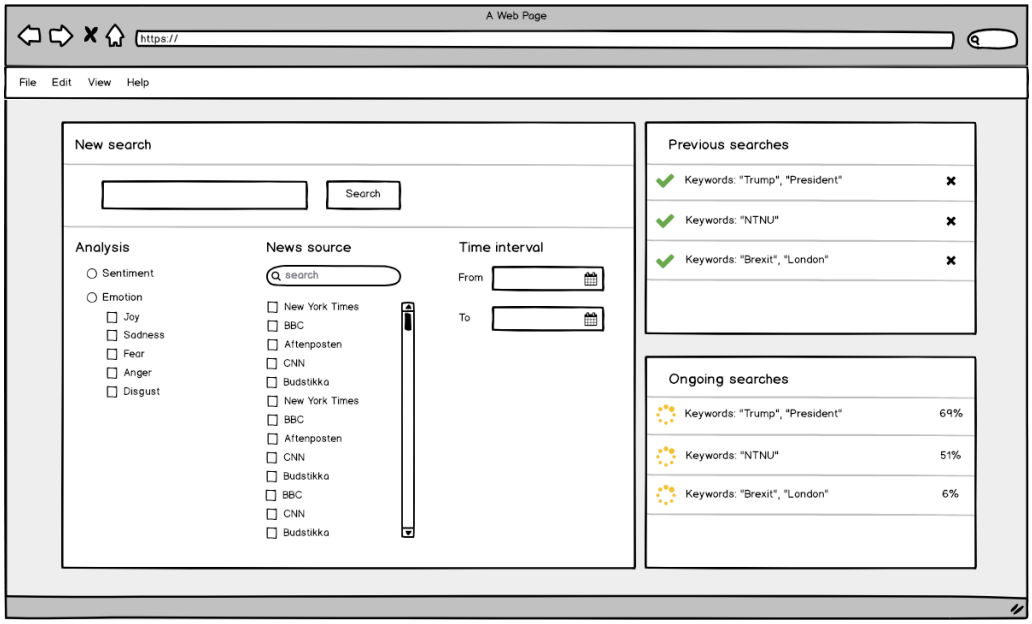
Gruppen har fra start diskutert mye om hvordan designet på applikasjonen burde se ut. Brukergrensesnittet har gått gjennom flere iterasjoner og endret seg mye fra første tenkte tanke til det vi endte opp med. Nedenfor skal vi se på hvordan produktdesignet har endret seg fra første *wireframe* til faktisk brukergrensesnitt ved levering av oppgaven.



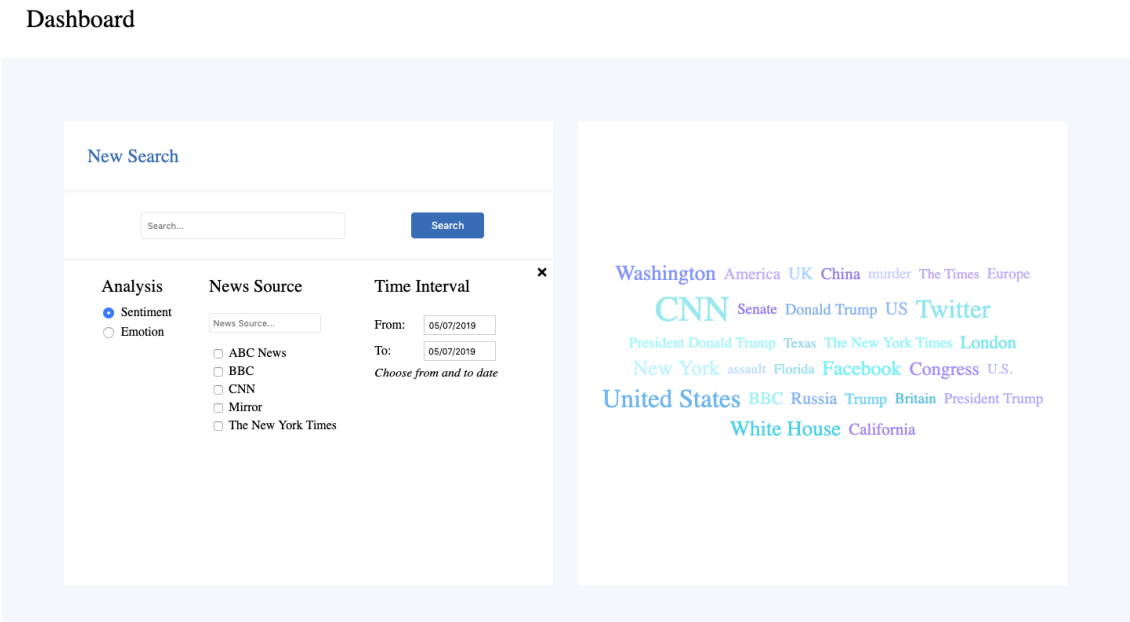
Figur 8: Første *wireframe* av Dashbordet



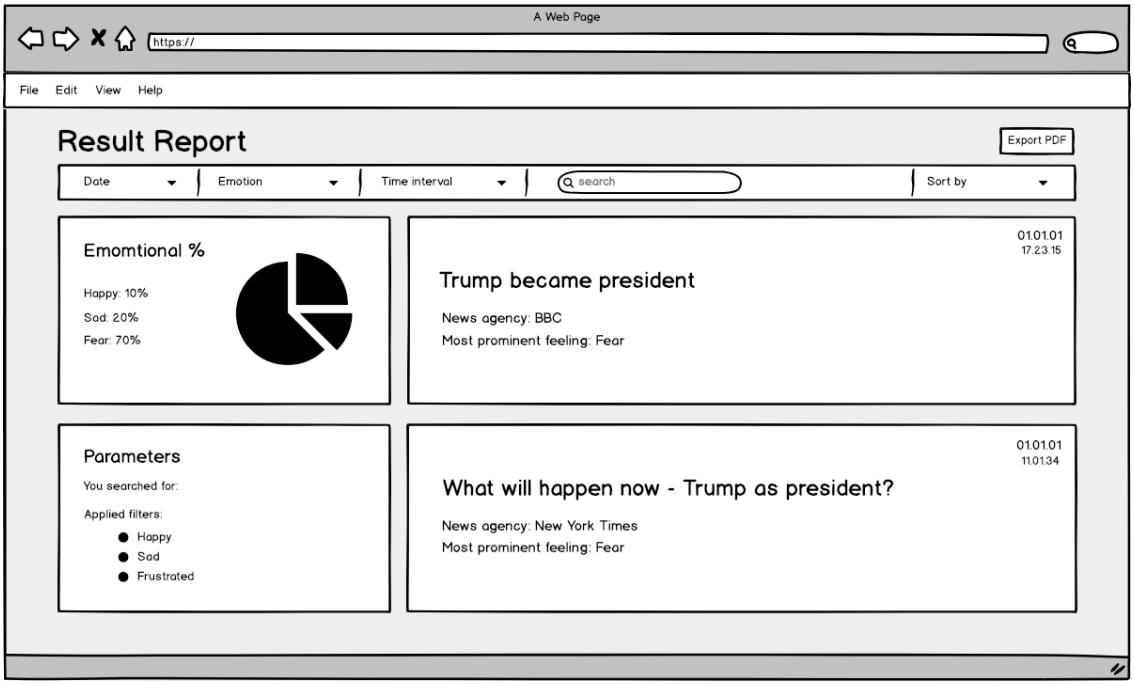
Figur 9: Skjermbilde av dashbordet, det første brukeren møter når en tar i bruk systemet



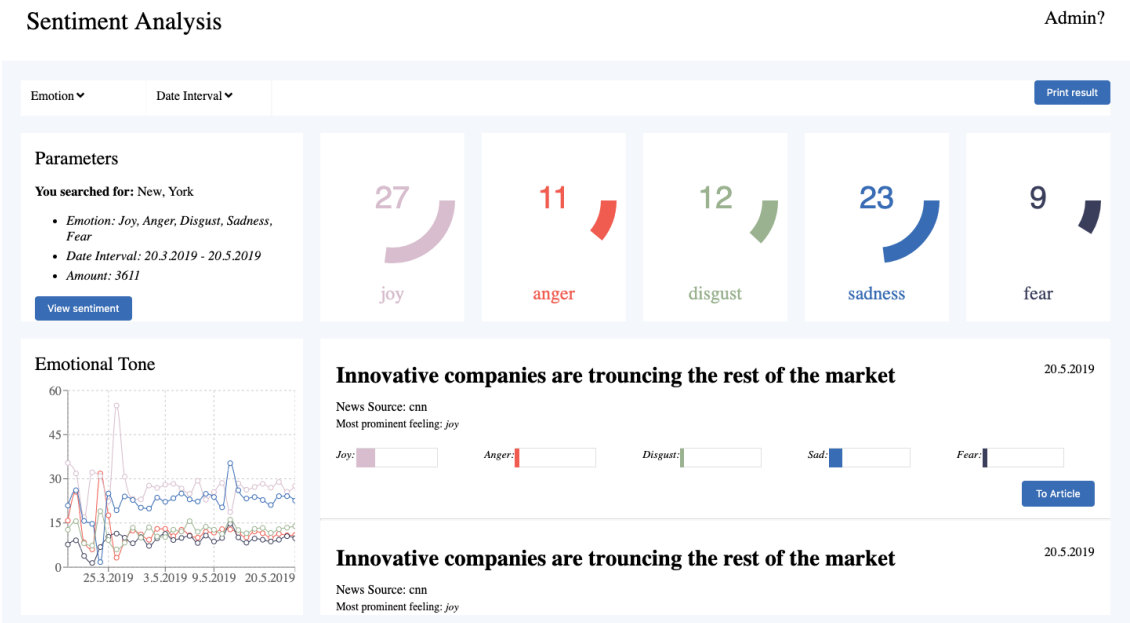
Figur 10: Første *wireframe* av mulighetene for å legge til filter før søk



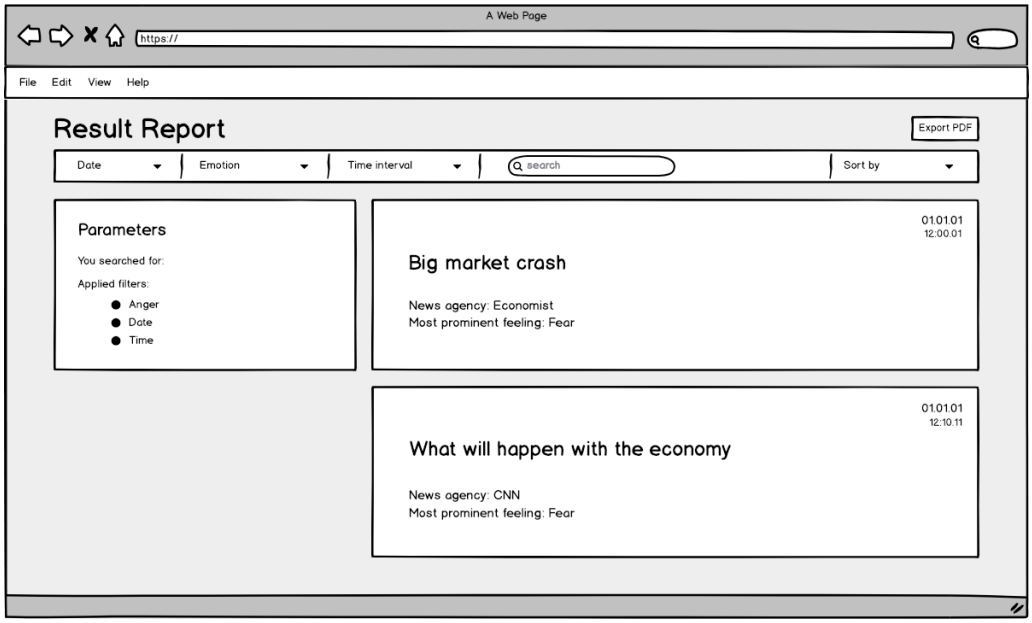
Figur 11: Skjermbilde av mulighetene for å legge til filter før søk



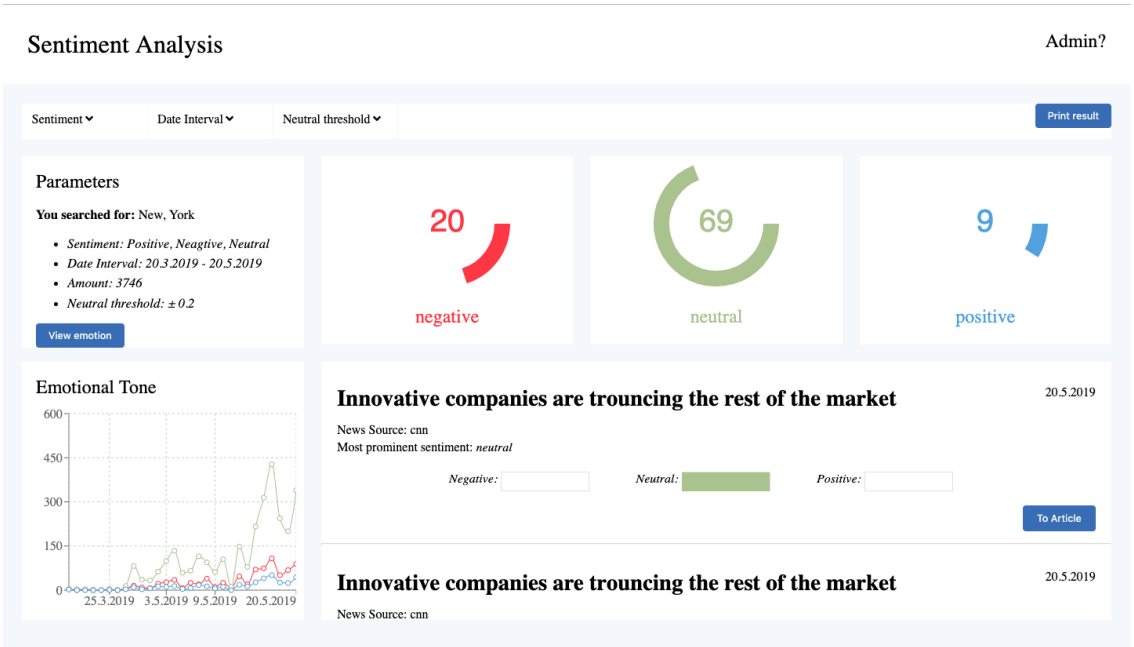
Figur 12: Første *wireframe* av resultatsude ved utført tonesøk



Figur 13: Resultatside ved utført følelsessøk



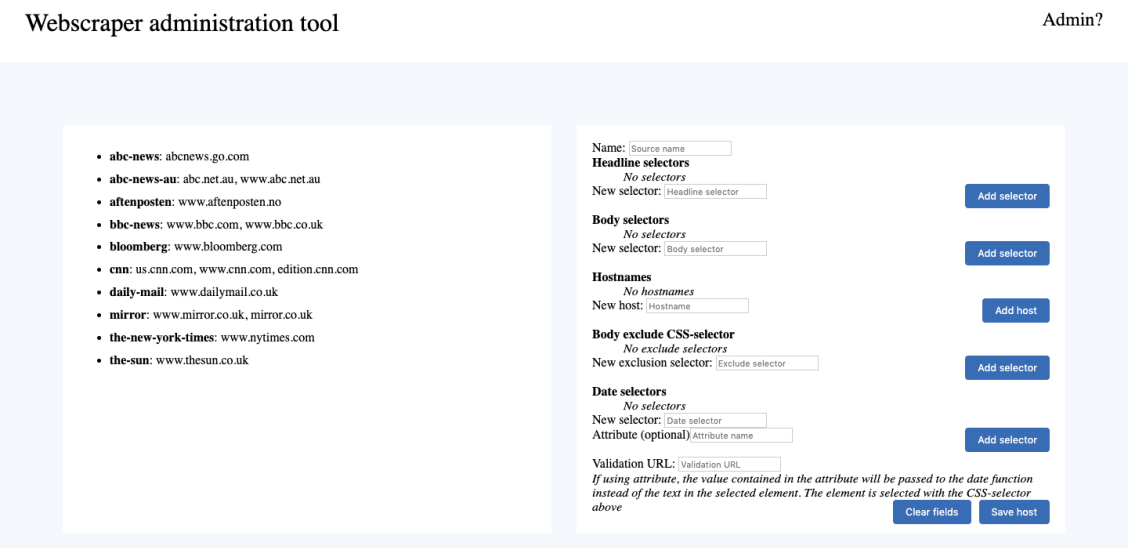
Figur 14: Viser første *wireframe* lagd av resultatside ved utført sentiment søk



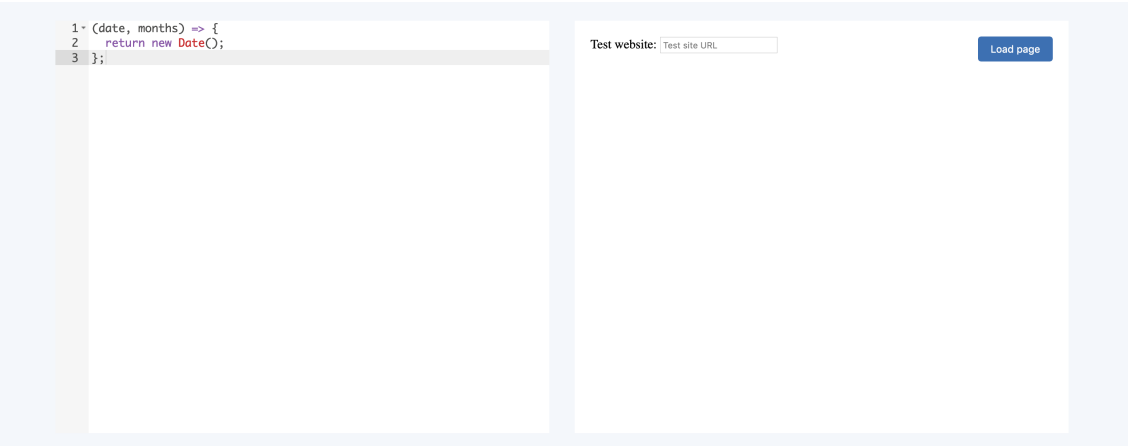
Figur 15: Viser resultatside ved utført sentiment søk

Tilleggsfunksjonalitet

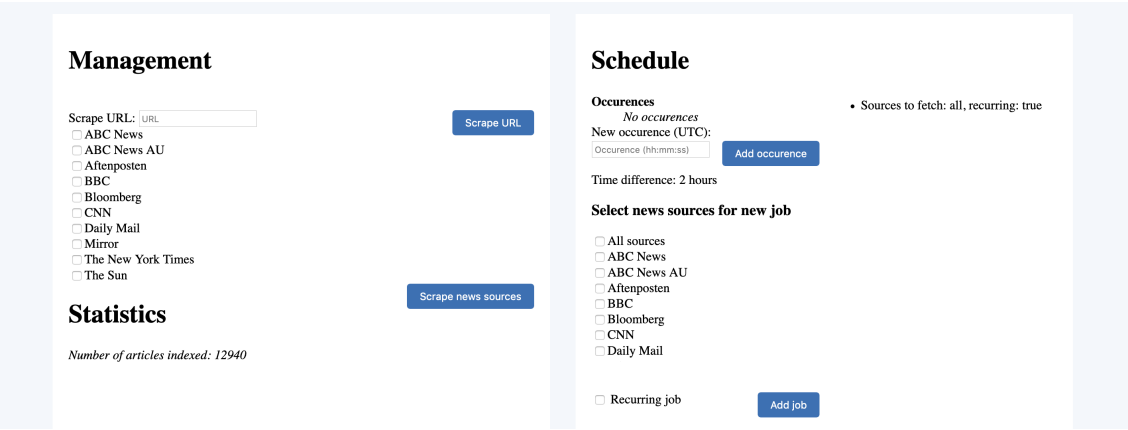
I denne seksjonen skal vi se på resultatene av tilleggsfunksjonalitet lagt til applikasjonen. Dette består av en administrator side for administrering av *Webscraperen*.



Figur 16: Viser administratorsiden, den øverste delen



Figur 17: Viser administratorsiden, den midterste delen



Figur 18: Viser administratorsiden, den nederste delen

4.3 Ingeniørfaglige resultater

I dette delkapittelet vil vi ta for oss de ingeniørfaglige resultatene. Disse er hentet fra visjonsdokumentet [82] og er basert på mål gruppen satte seg på starten av Bachelorprosjektet. Målene vi har valgt å beskrive her, er de vi mener er mest relevante for produktet og hvordan statusen på prosjektet var ved innlevering av bacheloroppgave 20.05.2019. Hvert mål vil bli tildelt en beskrivelse, samt en beskrivelse på hvordan det stod til under utviklingsperioden og ved levering.

4.3.1 Mål

1. *Systemet skal ha funksjonalitet for søking etter artikler og utføring av sentimentanalyse.*

   Hensikten med systemet er at en bruker skal kunne utføre en sentimentanalyse av ønsket ord eller frase på mediebildet i verden. En bruker kan nå utføre dette ved å enten skrive inn ord eller frase eller klikke på ord i ordsky. Systemet har funksjonalitet for søking etter artikler og utføring av sentimentanalyse på disse artiklene, der bruker vil få en overordnet oversikt over gjennomsnittlig statistikk på de ulike sentimentene og hvordan disse har vært daglig i tidsintervallet artiklene er hentet fra. Systemet har også funksjonalitet for å utføre en følelsesanalyse på de ulike artiklene søkt etter. Artiklene vil da bli analysert på følgende følelser: *Fear, Disgust, Joy, Anger og Sadness.*

2. *Bruker skal kunne eksportere resultatet av sentimentanalysen til PDF.*

   Systemet som leveres ved slutten av Bachelor perioden vil ha funksjonalitet som støtter muligheten for eksportering av resultatet av analysen til PDF dokument. Resultatsiden vil ha en *Skriv ut*-knapp som bruker kan trykke på, denne vil åpne et utskriftsvindu som lar bruker skrive ut resultatside, samt lagre det som en PDF fil. Bruker vil da kunne bruke analysen, skrive den ut eller vise den fram enkelt og oversiktlig uten og måtte gjennomføre analysen på nytt på siden, som da også krever Internett.

3. *Systemet skal ha en oppetid på 99.9%.*

   Systemet har, siden vi utrullet det for første gang, vært tilgjengelig og åpent på Internett uten nedetid. Dette hvis vi ser bort i fra eventuelt nedetid ved ny utrulling av applikasjon.

4. *Bruker trenger ikke opprette bruker eller lagre informasjon om seg selv for å få tilgang til systemet.*

   Eneste kravet systemet stiller til bruker er at en har tilgang til Internett. Bruker trenger ikke opprette profil eller lagre noe form for informasjon om seg selv for å få tilgang til systemet. Vi har derimot lagt til muligheten for en admin side, der en må opprette en profil på Auth0 for å få tilgang til *Webscraper Admin tool*, men dette kreves ikke for å få tilgang til systemet og søkefunksjonaliteten.

5. *Systemet skal alltid være åpent og tilgjengelig.*

   Systemet krever Internett for å kunne bli kjørt, da det er koblet mot eksterne ressurser som nåes via Internett. Systemet har, siden vi utrullet det, vært tilgjengelig og åpent uten nedetide, hvis vi ser bort i fra nedetiden ved ny utrulling av applikasjonen.

6. *Systemet krever lite vedlikehold.*

   Systemet vil i stor grad drifte seg selv og vil kreve lite vedlikeholde. Kildekoden har blitt dokumenter nøye, se vedlegg [83], og systemet vil derfor være lagt opp til å være lett å vedlikeholde ved behov.

7. *Skal være mulig å integrasjonsteste applikasjonen i senere tid.*

   De viktigeste funksjonene i systemets frontend har blitt integrasjonstestet, som blir beskrevet i neste avsnitt **Tester**. Disse har blitt dokumentert og lagt opp slik at det vil være lett å kjøre disse testene i applikasjonen i senere tid om ønskelig. Det vil være mulighet for å gjenskape testresultatene vi har kommet fram til.

8. *Applikasjonen skal ikke installeres på maskin eller mobile enheter, men vil være tilgjengelig på internett.*

   Applikasjonen stiller ikke noe krav til bruker om at den må lastes ned på maskin eller mobile enheter, men vil være tilgjengelig så lenge bruker har tilgang på Internett. Den vil kunne åpnes på både maskin og mobile enheter.

9. *Det stilles ingen krav til bruker ved ny utrulling av applikasjon.*

   Ved ny utrulling av applikasjon vil den fremdeles være tilgjengelig med oppetid under bygging og *staging*. Applikasjonen vil derimot ha en kort nedetid under *deployment* til selve produksjonsmiljøet. Ved utrulling stilles det ingen krav til bruker, og vedkommende vil ikke legge merke til utrullingen bortsett fra under den eventuelt korte nedetiden.

4.3.2 Tester

Under utviklingen bestemte vi oss for å fokusere på integrasjonstesting av applikasjonen vår, primært *backenden*. I dette delkapittelet skal vi ta for oss og vise hvordan vi kjører testene, samt vise hvor stor del av koden vi dekker. Figur 4.3.2 viser hvordan man kjører testen fra terminal.

```
SentimentAnalysis(master*) » npm test
```

Figur 19: Skjermbildet viser hvordan en kjører integrasjonstestene i terminalen

Figur 4.3.2 viser hvor stor del av koden som er dekket av test, og hvor stor prosentandel hver enkelt del av prosjektet dekker.

/ src / server

Files					Coverage
api	242	183	16	43	75.62%
routes	213	127	27	59	59.62%
ws	106	80	6	20	75.47%
app-env.js	10	0	3	7	0.00%
ics.js	16	0	3	13	0.00%
scheduler.js	288	47	104	137	16.32%
server.js	52	32	4	16	61.54%
Folder Totals (7 files)	927	469	163	295	50.59%
Project Totals (17 files)	927	469	163	295	50.59%

Figur 20: Skjermbilde viser hvilke deler av koden som er testdekket

4.4 Administrative resultater

I dette delkapittelet skal vi ta for oss de administrative resultatene. Disse er knyttet til hva som har blitt skrevet i prosjekthåndboka i forhold til Gantt diagrammet, framdriftsplanen. For å se oversikt over alle målene gruppen hadde satt seg se prosjekthåndboka, [80].

4.4.1 Prosjekthåndboka

Gantt Diagram

Gantt Diagrammet ble delt inn i oppstart og 14 ulike sprinter. Hver sprint tilhører enten Backend, Frontend, Dokumentasjon eller Maskinlæring, og de ulike rollene er fordelt på medlemmene i bachelorgruppen. Vi bestemte oss for et slikt oppsett ettersom det ville være enklere for hvert medlem å sette opp en egen sprint, med mål den personen hadde for en periode, framfor at flere måtte samarbeide om å sette sammen en sprint. Det er derfor noen sprinter overlapper hverandre.

4.4.2 Utviklingsmetodikk

Gruppen har brukt ScrumBan som utviklingsprosess. Vi har brukt sprintene satt opp i Gantt diagram som utgangspunkt for sprintene i ScrumBan. Målene satt opp i hver enkelt sprint, ble overført til Trello og ble skrevet opp som produktets *Backlog*, se 3.5. Ettersom sprintene i Gantt diagramme er delt inn og satt opp basert på de ulike rollene innad i teamet, måtte vi slå sammen sprinter for at de skulle inneholde arbeid i *frontend* og *backend*. Enkelte sprinter fra Gantt diagrammet går derfor over flere sprinter i ScrumBan. I tabellen nedefor illustrerer vi hvordan vi har satt sammen og fordelt de ulike sprintene fra Gantt over til ScrumBan.

ScrumBan Sprint	Gantt-diagram Sprint
Sprint 1	Sprint 1 & 2
Sprint 2	Sprint 3 & 6
Sprint 3	Sprint 4 & 7
Sprint 4	Sprint 4 & 8
Sprint 5	Sprint 5

Tabell 12: Tabellen viser oversikt over hvordan sprintene i Gantt ble satt sammen til sprinter i ScrumBan

I Prosjekthåndboka er det lagt med gantt diagrammet vi benyttet og statusoppdateringer på hva som har blitt gjort, skrevet av hvert medlem for hver enkelt uke gjennom bachelorprosjektet. Gjennom disse kan en se når oppgaver og mål faktisk ble fullført i forhold til hva som var planlagt. Utviklingsprosessen kommer også fram her da statusoppdateringene informerer om hvilke mål som ble utført når, og Gantt diagrammet informerer om fullførte og ikke fullførte sprintmål.

5 Diskusjon

Diskusjonsdelen vil først ta for seg utfordringer innen forskningsdelen på sentimentanalyse og reflektering over resultater oppnådd. Deretter diskuteres vitenskapelige resultater knyttet til systemutviklingen, ingeniørfaglige resultater, evaluering av arbeidsprosess og utviklingsmetodikk, helhetlig systemperspektiv og gruppearbeidet.

5.1 Maskinlæring

Først vil vi diskutere generelt om utfordringer rundt sentimentanalyse ved å bruke engelske modeller som utgangspunkt. Vi tar for oss flere problemstillinger knyttet til sentimentanalyse som også gjelder for norske modeller. Deretter tar vi for oss utfordringene som oppstod ved å finne et ferdig norsk datasett, som resulterte i å annotere et eget. Til slutt vil vi diskutere ulike modeller brukt til sentimentanalyse.

5.1.1 Engelsk sentimentanalyse

Det finnes flere ferdige modeller for sentimentanalyse på engelsk, ettersom det er et av verdens mest brukte språk. Felles språk har den fordelen at flere kan gjenbruke andre sine produseringer og utnytte ferdig utviklet teknologi. Det finnes ferdige modeller som analyserer sentiment og tone utviklet av store selskap, slik som IBM. Nevnte har fått oppmerksomhet for superdatamaskinen Watson. Den er i eliteklassen innen maskinlæring og er en god representasjon av ferdige maskinlæringsmodeller.

IBM Watson NLU presterte med 85,7% nøyaktighet på 30 000 engelske anmeldelser fra Amazon. Om modellen estimerer sentimentet til 100 anmeldelser, vil 14 være feilestimert. Anmeldelsene var kun annotert med positiv eller negativ (binærklassifisering). Dersom modellen hadde estimert tilfeldig, ville nøyaktigheten vært 50%, som forteller at modellen gjør det bedre enn ved tilfeldig utvelgelse. Det er mulig det kunne vært både bedre og dårligere resultat dersom vi hadde valgt ut 30 000 anmeldelser fra Amazon testsettet annerledes, selv om rekkefølgen på anmeldelsene er mikset i forkant.

Færre kategorier gjør det lettere for modeller å få høy nøyaktighet ettersom det er større sannsynlighet for å gjette riktig og mindre nødvendig lære seg å skille mellom kategorier. Det er kun brukt to kategorier i Amazon datasettet og vi fant flere andre engelske datasett brukt til sentimentanalyse med samme trekk. Amazon datasettet vil derfor være godt nok til å gi en pekepinne på kvaliteten til modellen.

For å sammenligne IBM Watson NLU med en annen modell, ble det valgt å bruke fastText. Nevnte kan trene raskt på store datasett, men likevel prestere like godt som mer avanserte modeller. Amazon datasettet består av 4 millioner anmeldelser og anses som et stort datasett. FastText ble ferdig trent på omtrent 12 minutter, med treningsettet som bestod av 3,6 millioner anmeldelser. På den korte tiden fikk modellen 93,8% nøyaktighet på testsettet, som er 8.1% høyere enn Watson. Trening-test-fordelingen til datasettet er 99/1 for fastText modellen. Denne skiller seg ut fra den vanlige fordelingen på 80/20. Å bruke et lite testsett kan gi feilaktig høy nøyaktighet, som John Arne Øye konkluderer med i sin

masteroppgave (fra kapittel om tidligere arbeid 2.1.1).

IBM Watson NLU og fastText trent på Amazon klarer å estimere de fleste anmeldelser riktig. Mer kompliserte anmeldelser, slik som den positive anmeldelsen "*HE slips a little but still a well worth waiting conclusion!: Not as great as every other book but wow what an ending!*" (fra resultat 4.1.1), blir vanskeligere å estimere. Watson estimerte anmeldelsen som 29,8% negativ og fastText estimerte den positiv med 100% sikkerhet. Vi kan anta delene "*HE slips a little*" og "*Not as great as every other book*" gjorde at Watson estimerte feil.

Det er ikke bare Watson NLU som kan estimere feil. FastText estimerte anmeldelsen "*I thought is wasn't good.: This is about a boy and a cricket. Mario, the boy, is working at a newsstandwhen he hears a noise. He looks under a trash can and finds a cricket. He buys a cage for the cricket, and that's how it turns out. The part I liked is when the cricket chirps and asks the man who wasgoing to steal the bell what he was doing. I did not like the rest*" som positiv med 95% sikkerhet når den er annotert som negativ. Watson estimerte at anmeldelsen var 67% negativ. Det er mulig fastText estimerte feil på grunn av "*The part I liked is when...*" og på grunn av ordet "*not*", som kan snu sentimentet i setningen.

Kjennetegnet på anmeldelser er at de ofte er korte og fylt med sentiment. Spesielle kjennetegn kan gjøre at modellen overtilpasser seg dataene (overfitting). Det blir oppdaget ved at nøyaktigheten på treningssettet er høyere enn nøyaktigheten på valideringssettet. Hos fastText får vi ikke sett nøyaktigheten på valideringssettet fordi treningen foregår internt i modellen. Dersom testsettet består av samme kjennetegn som treningssettet vil det ikke bli oppdaget. At modellen blir spesialist på anmeldelser kan resultere i at fastText trent og testet på samme datasettet gir høyere nøyaktighet.

IMDb datasett

FastText ble trent med et annet engelsk datasett, IMDb, som består av 50 000 anmeldelser annotert med positiv og negativ. Treningen tok mindre enn halvparten av tiden til Amazon datasettet, fordi IMDb datasettet er mindre enn Amazon. Nøyaktigheten ble 82.0% på testsettet med samme Amazon anmeldelser brukt til å teste modellene over. IMDb anmeldelser er derimot også anmeldelser, som gjør at modellen blir spesialisert og kan få bedre nøyaktighet på testsettet.

Størrelsen til treningssettet er 50 000 anmeldelser og testsettet er 30 000 anmeldelser. Det betyr at trening-test-fordelingen er 63/37, som ikke er den vanlige fordelingen på 80/20. Dersom det å bruke et lite testsett gir bedre nøyaktighet (2.1.1), betyr det også at små treningssett gir dårligere nøyaktighet. Et treningssett på 50 000 anmeldelser klassifiseres ikke som et stort datasett og kan gi dårligere nøyaktighet samt at det ikke gir fastText nok treningsdata.

Dersom vi hadde testet modellene på store dokument som ikke var anmeldelser, har vi grunn til å tro at Watson hadde prestert bedre enn fastText modellene. Èn god nøyaktighet kan skjule mye og nøyaktigheten på et testsett er ikke nok til å fortelle om en modell er god. I praksis vil vi at en modell skal kunne forstå alle typer dokument og analysere riktig på mest mulig. Det er det Watson har gjort og er ikke er spesialisert for anmeldelser.

Hvordan man bør implementere en modell avhenger av hva den skal brukes til. Dersom målet er å analysere sentimentet på nye anmeldelser uten vurdering og man har tilgang til et datasett av samme type, vil det å lage en spesialtilpasset modell være en god avgjørelse. Men en modell slik som IBM Watson NLU prøver å lage en generalisert modell for alle typer data.

Modellene for engelsk sentimentanalyse presterte godt. Siden vi ønsker å utføre sentimentanalyse på norsk, ville det vært enklere å oversette de norske dataene til engelsk og deretter analysere med en engelsk modell. Problemet med denne tilnærmingen er at setningsoppbyggingen til det engelske og det norske språket er ulikt og vi vil kunne miste sentiment under oversetting av teksten. I delen om tidligere arbeid (2.1.1) konkluderer de med at å oversette ord for ord blir for unøyaktig.

5.1.2 Ferdig norsk datasett

For å lage en modell for sentimentanalyse på norsk må man først finne et stort datasett. Det er en av de mest krevende oppgavene for maskinlæring på norsk språk fordi det er vanskelig å finne et åpent, gratis, menneskelig annotert datasett for sentiment. Underveis fant vi NoReC, som ble laget i 2018 av UiO på grunn av mangel på norske datasett. Det er totalt 35 200 anmeldelser i form av artikler i NoReC. Det er et stort datasett med tanke på lengde, men ikke på antall eksempler. Vi tester med 7 031 eksempler, og får nøyaktigheten 72,0% på litt under 1,5 minutter.

En stor feilkilde i datasettet er at hvert eksempel er langt, med kun ett sentiment. En anmeldelse er bygd opp av en innledning, en fortelling av innholdet i produktet og en refleksjon, med både positive og negative vurderinger av produktet. Fortellingen av produktet blir inkludert som feature til dataeksempelet. Det betyr at en positiv anmeldelse kan inneholde mye negativt sentiment i fortellingen om produktet. Når terningkastet for anmeldelsen enten 3 eller 4, blir det klassifisert som nøytralt. Utfordringen er at terningkastene henholdsvis betydde middelmådig og god, som er følelsesladde ord.

For å få et godt datasett, er det også viktig med jevn fordeling over de ulike kategoriene. Datasettet basert på NoReC har nesten 3000 negative, 17500 nøytrale og 15000 positive anmeldelser. Det er under 13% negative - en verdi som skulle vært 33,3%. En mangel på negative anmeldelser gjør at modellen har sett flere, og blir bedre tilpasset, positive og nøytrale anmeldelser. En løsning på fordelingen er undersampling, en metode som tilfeldig fjerner positive/nøytrale anmeldelser fra datasettet til det er 3000 av hvert sentiment. Ulempen er at datasettet blir enda mindre enn det allerede er. Vi valgte derfor ikke å benytte oss av metoden.

Ved å fjerne nøytralt sentiment fra datasettet, satt vi igjen med et testsett med 3589 anmeldelser, hvorav 461 er negative (12,8%). Resultatet av å trene med fastText er en nøyaktighet på 94,4%. Det er en høy nøyaktighet som slår de engelske modellene. På grunn av partiskhet kan vi ikke konkludere med at denne modellen er bedre. Siden 12,8% var negative anmeldelser, vil det å estimere bare positiv til alle anmeldelsene i testsettet gi en nøyaktighet på 87,2%. De andre datasettene er jevnt fordelt og vil ikke ha denne feilkilden. NoReC vil ikke være tilfredsstillende nok til å kunne gi en oversikt over trender i nyhetsbildet ved å bare kunne estimere rundt 2/3 riktig. Dermed gikk valget til å finne et annet datasett.

5.1.3 Eget norsk datasett

Vi besluttet å annotere et eget datasett ettersom det var få norske datasett for sentimentanalyse å finne åpent på nett. Å menneskelig annotere datasett går ut på å selv bestemme hvilket sentiment tekstlig data får. Det er både en tidkrevende prosess og vanskelig fordi det ikke alltid finnes et bestemt svar. Derfor fant vi en metode som annoterte automatisk.

Først må man ha store mengder tekst. En av de største åpne APIene i verden er Twitter API. Twitter er

et nettsted for å ytre sine meninger i korte tekster, som betyr at tweetene ofte inneholder mye sentiment. Ved å annotere på ord og emojier, ble det fort et stort datasett med flere tusen eksempler. Hvert eksempel er derimot veldig kort, altså motsatt av NoReC datasettet. Datasettet ga til slutt en nøyaktighet på 81.6% med omtrent like lang treningstid som NoReC, men med en økning i nøyaktighet på 10%.

På samme måte som for modeller trent på anmeldelser, kan denne modellen spesialisere seg på tweets. Det kan gi en høyere nøyaktighet enn for andre typer dokument. Dersom modellen ble laget for å analysere tweets, er ikke dette et problem man skal tenke på.

For det andre kan automatisk annotering basert på enkeltord og emoji i teksten gi feil sentiment, for eksempel ved at en tweet inneholder ordet *"ikke"*. Derfor kan en tweet ha motsatt sentiment av det de fleste ordene i tweeten antyder. En annen tekstlig feilkilde, som er mye kjent innen tekstklassifisering, er sarkasme og ironi. Dersom man hater en film, men sarkastisk sier den er god, vil automatisk annotering annotere den som en positiv anmeldelse. Dette gjelder også for trente modeller i maskinlæring som ønsker å evaluere en tekst som er sarkastisk (fra teori om sarkasme i NLP, 2.2.2).

Twitter returnerte tweets kategorisert som norske. Det ble raskt oppdaget at mye ikke var norsk. Derfor måtte tweetene filtreres enda en gang ved å sjekker om tweeten inneholdte ett eller flere norske ord ved hjelp av norske ordlister. Det gjorde at antall tweets gikk ned, som betyr at det var flere tweeter som ikke inneholdte et eneste norsk ord. I tabell 13 er tweetene fra treningssettet preprosessert og emojier fjernet. Vi mener at tweet nr. 2, 5 og 6 er annotert feil. Tweet nr. 1, 3, 4 og 7 ser ut til å være korrekt annotert.

Nr.	**Kategori**	**Tweet**
1	Nøytral	&gterrf&gtfye
2	Nøytral	mamma er håpløs med elektroniske ting jeg har måttet hjelpe henne flere ganger med å justere volum på mobilen henn
3	Nøytral	pyton skaperen rolf håndstad bedre kjent som rhesus minus er død år gammel
4	Positiv	denne gjengen har valgt det feteste band navnet vi har hørt på leeenge sjekk ut deres første låt nå på spotify
5	Positiv	synes du skal holde deg til sport og analyser om det ingenting interessant om støres tale syntes d
6	Negativ	tror ikke nødvendigvis at det er en dum ting å være opptatt av
7	Negativ	brutalllll

Tabell 13: Tweets fra resultatdelen

En feil i norskfiltreringen er at det kun blir sjekket om sekvensen av bokstaver finnes i tweeten. Siden *"er"* regnes som et norsk ord i en av ordlistene, vil alle ord som inneholder *"er"* beholdes da det regnes som en norsk tweet. Vi ser denne feilen fra tweeten *"&gterrf&gtfye"*, som ganske tydelig ikke er norsk. Siden tweets ofte har skrivefeil, vil det å filtrere ved bruk av hele ord utelukke mange tweets. En annen tweet også nevnt i resultatdelen er *"brutalllll"*. Tweeten ble inkludert i vår filtrering siden sekvensen av bokstavene *"brutal"* er i tweeten, men det er ikke frittstående. Det kreves en mer avansert filtrering for å gjøre det nøyaktigere. Tweetene i avsnittet inkluderer ikke emojier og er etter preprosessering. Et forslag til forbedring er å fjerne ordene som kan finnes som en del av ikke-engelske ord i ordboken vi bruker i filtreringen.

Det er lett å glemme å fjerne emojiene etter annoteringen. Dersom emojiene blir beholdt, kan modellen finne igjen mønstrene brukt til å annotere datasettet og enkelt finne korrekt sentiment til tweetene. Dette gir uvanlig høy nøyaktighet som ikke alltid er lett å oppdage. Derimot må vi beholde de ordene som ble brukt som en feature til å annotere datasettet. Modellen kan ha funnet dette mønsteret og derfor fått høyere nøyaktighet. Det vi vil er at modellen finner sentiment ved å finne mønsteret i sekvensen av ord.

Binærklassifisering ved å fjerne nøytralt sentiment ga nøyaktigheten 95,3%. Den er den høyeste binærklassifiseringen vi fikk i rapporten. Derimot kan modellen være partisk ettersom ordene vi bruker for å annotere tweetene ikke fjernes i etterkant. Fordelingen av sentiment er 50/50 i trening og testsettet som gir modellen et utgangspunkt på 50% ved å tilfeldig estimere eller kun estimere på en av kategoriene. Det er også fortsatt rundt 110 000 eksempler i datasettet, som skal være stort nok til å kunne si at datasettet er godt eller dårlig.

5.1.4 Stemming

Stemming av tekst skal i utgangspunktet gi bedre resultat i form av nøyaktighet i følge flere referanser (fra kapittel om tidligere arbeid, se 2.1.5). For tweetene i datasettet ble det ingen forskjell og nøyaktigheten ble 79,3%. Det er 2% dårligere enn det ustemte settet. Dette er ikke en veldig betydelig forskjell og dermed kan vi ikke med sikkert si at stemming gjorde datasettet dårligere.

Stemmingen skjedde i etterkant av annoteringen, og hver tweet har samme sentiment før og etter stemming. En fordel med stemming, annet enn bedre nøyaktighet, er at flere ord blir like og vi får plass til flere ord i ordforrådet til modellen. Det betyr at den i teorien skal kunne forstå flere ord. Det er viktig å tenke på at modellen kun vil forstå stemte ord om den har trent på stemt datasett, og at teksten må stemmes dersom den skal analyseres av modellen. Til fordel skjønner modellen nå flere varianter av samme ord.

En ulempe med stemming er at det kan ødelegge sentimentet i tekst. I engelsk stemming kan to ord som i utgangspunktet har forskjellig sentiment bli stemt til samme ord. Et eksempel er at det nøytrale ordet "meaning" blir til "mean" og plutselig får negativt sentiment. Det samme kan også forekomme på norsk; Det negativt ladde ordet "merkelig" blir det nøytrale ordet "merk". En fordel med stemming er at datasettet blir mer komprimert og inneholder mer informasjon per bokstav i forhold til et ustemt datasett. Stemming gjorde datasettet vårt 9,5% mindre i størrelse og trente også 9% raskere med samme parametre som det ustemte datasettet.

5.1.5 Mikset datasett

Tweets inneholder mye slang og skrivefeil som ikke er en del av det norske språket. Modellen bør kunne estimere tekst på nett, uansett hvilken form og kvalitet på språket. For å kunne analysere alle typer dokumenter godt, må modellen bruke data som også har god grammatikk, slik som artikler skrevet av journalister. NoReC er et datasett med god grammatikk, men denne ikke presterte så godt alene. Vi så deretter om å mikse datasettene ville gi en god presisjon.

Datasett	NoReC	Ustemt Twitter	Mikset
Nøyaktighet	72,0%	81,6%	77,7%

Tabell 14: Nøyaktighet mikset datasett

Tabell 14 viser en oversikt over nøyaktigheten til NoReC, det ustemte datasettet og mikset datasett (fra resultat 4.1.3). Mikset datasett vil ikke gi bedre nøyaktighet enn det ustemte datasettet basert på Twitter. Det er derimot ikke en god nok grunn til å si at modellen trent på ustemt datasett gir et bedre resultat enn modellen trent på mikset datasett. Det er fordi den kombinerte modellen kan være generalisert, slik som IBM Watson, og gjør det bedre på ulike typer dokumenter enn modellen spesialisert for Twitter.

5.1.6 Valg av hyperparametre

Hyperparameterne brukt til trening av modell i fastText er basert på mange ganger prøving og feiling. Det er slik man finner ut hvilke hyperparametre som passer best for det datasettet du gir modellen. Derfor vil det ikke være sikkert vi har funnet de beste parametrene som finnes, men det er de vi har funnet så langt.

5.1.7 Modeller for sentimentanalyse

Modellene for sentimentanalyse brukt i denne rapporten er IBM Watson NLU, fastText og et LSTM nevralt nettverk. Det viser seg at modellene presterer dårligere på de norske datasettene.

Modell	**Amazon (eng.)**	**IMDb (eng.)**	**NoReC**	**Stemt**	**Ustemt**	**Mikset**
fastText	93,8%	81,9%	70,5%	79,1%	81,5%	77.7%
Watson NLU	85,7%	-	-	-	-	-
LSTM ett lag	-	-	-	-	78,4%	-

Tabell 15: Nøyaktighet ulike modeller for sentimentanalyse

Hyperparametre brukt til å trene LSTM modellen er gitt i tabell 16 og har blitt bestemt ut i fra prøv-og-feil metoden samt kunnskap om ustemte datasettet. Vi bestemte oss for 6 epoch ved å se når modellen begynte å *overfitte* dataene og avslutte antall iterasjoner der. 'Ordliste 30000" betyr at ordlisten til modellen består av 30 000 ord. Det er 64 LSTM nevroner i det skjulte laget, med 60% dropout (som er høyere enn normalt) og 20% recurrent dropout. Datasettet brukt til trening bestod av omtrent 137 000 anmeldelser som hadde kategorien positiv, negativ eller nøytral. LSTM brukte 6 minutter på treningen på den virtuelle maskinen, mens fastText brukte 45 sekunder på en Macbook Pro (beskrevet i

Læringsrate	Epoch	Embedding	Ordliste	Units	Dropout	Recurrent Dropout	Batch
0,001	6	8	30000	64	0,6	0,3	256

Tabell 16: Hyperparametre LSTM

teknologidel 3.3.5). Nøyaktigheten til LSTM ble 78,4% mot fastTexts 81,6%. FastText er kjent for å få høy nøyaktighet på kort tid på store datasett. Dersom vi hadde brukt et mindre datasett, er det mulig LSTM ville prestert bedre.

Hyperparameterne brukt i LSTM modellen kan alltid forbedres. Det som avhenger er hvor god tid man har til å optimalisere nøyaktigheten. Verdiene vi brukte var de beste vi fant på tiden vi hadde tilgjengelig.

5.2 Vitenskapelige resultater

Produktet består av funksjonalitet prosjektgruppen mener vil være mest relevant og nyttig for tenkt målgruppe av brukere av systemet. I dette delkapittelet skal vi vurdere valg av funksjonaliteten i systemet og eventuelle svakheter.

5.2.1 Produkt

Valgt funksjonalitet reflekterer systemets tenkte målgruppe av brukere. Vi ønsket å lage en applikasjon som gjør det enkelt å gjennomføre en sentimentanalyse på ønsket ord eller frase, med filtrerings muligheter som er vurdert som mest relevant. Ved å ha et fokus på å lage en applikasjon med et enkelt brukergrensesnitt, kan det benyttes av brukere uten teknologisk erfaring. Ved å tilby enkel funksjonalitet slik som filtrering av søk og sortering av resultater, vil ikke siden virke komplisert og en bruker vil enkelt og intuitivt forstå hva han eller hun kan gjøre i systemet. En svakhet her er dersom bruker har mye erfaring med navigering på nettsider, har det ikke blitt lagt til rette lagt for tastatur snarveier. Et søk kan utføres ved å taste på Enter-knappen, men dette er eneste snarvei som har blitt lagt til så langt. Et forslag kan være en snarvei for å bytte mellom sentiment og følelsessøk eller nullstille valg og starte et nytt søk.

Produktet tilbyr også utskrift av siden, slik at resultat kan eksporteres og benyttes videre uten å være koblet til internett. Først var planen at det skulle være mulig å eksportere resultatene til en PDF fil, i et ønsket format, slik at bruker kunne få lagret resultater utenfor systemet. Etter flere forsøk fant vi ingen god løsning som ga en god presentasjon i PDF format. Dette resulterte i at vi valgte en løsning der bruker kan skrive ut resultatet. Valget innebærer imidlertid at hele skjermbildet blir skrevet ut. Elementer som navigasjonsbar og knapper bli med, noe som er urelevant for bruker. Dersom bruker ikke er erfaren med å printe ut nettsider, kan det være en ikke er kjent med hvordan en PDF opprettes gjennom utskriftsvinduet. En videreutvikling av systemet bør inkludere muligheten for eksport av resultatene fra analysen.

I utgangspunktet måtte bruker av systemet velge om de ville gjennomføre en sentiment- eller toneanalyse før et søk ble utført. Men etter en nærmere vurdering av brukerens behov, så vi at det ville være mer hensiktsmessig for bruker å skifte mellom sentiment- og toneanalyse etter at søket har blitt gjort. Bruker slipper å gå tilbake, legge på filter og skrive søkeord på nytt, dersom en ønsker å gjennomføre en ny analyse eller har trykket feil. Denne løsningen vil også være hensiktsmessig med tanke på ordsky der det vil bli gjennomført en sentimentanalyse ved klikk på ord. Funksjonalitet for dette ble derfor lagt inn i form av en knapp, der bruker kan skifte mellom de to ulike typene analyse. Et problem som har oppstått her er tiden det tar fra knapp er trykket på til siden er oppdatert med ny data. Vi har forsøk å sette inn en form for *vente GIF*, men på grunn av måten vi har implementert resultatsiden på blir ikke denne vist før etter resultatsiden har oppdatert seg med ny data. Bruker vil derfor ikke få noe tilbakemelding på at knapp er trykt på og at ny data lastes ned. Dette er en svakhet som også strider mot designprinsipper for interaksjonsdesign.

I tidligere arbeid (2.1.3) har vi referert til en forskningsrapport, der aksjemarkedet har blitt studert og analysert og hvordan markedet har blitt påvirket av tweets gjennom bruk av sentimentanalyse. Vårt produkt består av et grensesnitt opp mot IBM sin ferdig trente modell for sentimentanalyse. Bruker trenger derfor ikke sette seg inn i verken maskinlæring eller programmering for å lage en modell for å utføre en analyse. Studering av aksjemarkedet og hvordan dette kan bli påvirket av informasjon formidlet gjennom ulike sosiale medier er ett eksempel på hvor vårt produkt kan brukes. Forskere kan

enkelt søke etter ord eller fraser knyttet til et selskap i aksjemarkedet og se hvordan dette er omtalte i media gjennom sentimentanalysen, forstå og sammenligne disse med utvikling i aksjekursen.

5.2.2 Design

I dette avsnittet skal vi ta for oss utformingen av brukergrensesnittet og hvilke valg vi har måtte ta underveis. Nedenfor følger en oversikt over de tre ulike sidene i webapplikasjonen og *wireframesene* knyttet til disse. Diskusjonen vil gi innblikk i endringene fra første *wireframe* til brukergrensesnitt ved levert produkt.

1. *Dashbord*

   Den første *wireframen* av Dashbordet hadde et søkefelt, samt oversikt over pågående søk og en oversikt over tidligere søk. Men etter at vi endret *backenden*, slik at analysen av artiklene allerede var gjennomført og lagret i databasen før en kan søke i de, ble det unødvendig å ha et pågående søkefelt ettersom et søk vil ta under 1 sekund å gjennomføre. Vi konkluderte også med at det ble unødvendig å ha en liste over tidligere søk, da det å gjennomføre et søk på nytt tar så kort tid. Framfor å ha et dashbord kun bestående av et søkefelt, ønsket vi å legge til noe som visuelt kunne vise hvilke ord og fraser som er mest nevnt i mediebildet. Dette er statistikk vi mener kan være interessant for en bruker for enkelt å få et visuelt inntrykk av medieomtale. I tillegg til å se hvilke type ord og fraser en kan søke på. En svakhet her er at ved klikk på ord, vil det uansett bli gjennomført en sentimentanalyse. Det ble derfor lagt til en knapp på resultatside, så bruker kan skifte til tonesøk etter utført sentimentsøk. Dette gir bruker økt fleksibilitet på allerede gjennomført søk.

2. *Resultat: Følelse*

   Det første vi så for oss når vi designet brukergrensesnittet her var at nyhetsartiklene skulle være fokuset på siden, at det var dette brukeren var interessert i. Som en kan se på wireframe, se 4.2.2, var 2/3 av resultatsiden oversikt over artikler tilknyttet søkeord, mens 1/3 viste fram søke parameterne og resultatet av analysen. Etter nærmere vurdering kom vi fram til at fokus i brukergrensesnittet må være å få fram statistikk som viser trender i mediebilde. Dette innebærer å se gjennomsnittlig statistikk i hele mediebildet, og ikke ha så stort fokus på hva en enkelt artikkel mener om en sak. Brukergrensesnittet ble derfor endret slik at statistikken er i hovedfokus, samt at vi inkluderte en *Emotional Tone* graf. Denne hadde som formål å vise hvor mange mangle artikler per dag i et gitt tidsintervall som omhandler hvert sentiment/følelse. Bruker kan også klikke på et punkt, en dag, i grafen og få analyse utført for denne dagen. Ettersom fokuset på enkelt artikkelen forvant, konkluderte vi også med at det ville være unødvendig og urelevant for bruker å søke etter spesifikke ord i titlene på artiklene. Vi valgte derfor å ikke implementere dette og fjerne det fra brukergrensesnitt, se 4.2.2.

3. *Resultat: Sentiment*

   Samme endring ble gjort her som over. I den originale *wireframen* ble det ikke vist noe statistikk på hvor sto prosentandel av mediebildet som var positivt, negativt og nøytralt. Dette ble lagt til som grafer, på samme måte som med følelsessøk. Grunnen til endringen er samme som nevnt i avsnittet over.

Vi skal nå se videre på brukergrensesnittet og hvordan dette er utformet ved hjelp av ulike designprinsipper, samt svakheter ved det.

Under utviklingen av systemet lå hovedfokuset på å forenkle og tilgjengeliggjøre sentimentanalyse for forskere. Vi ønsket å lage en applikasjon som var enkel og intuitiv å bruke. For å oppnå dette benyttet vi oss blant annet av *Don Norman's Principles of Interaction Design*. Ved å ha fokus på å synliggjøre funksjonalitet og muligheter brukeren har, samt sørge for at bruker alltid får en tilbakemelding etter utført handling, har vi også sørget for at bruker er klar over begrensninger systemet har. Dersom bruker prøver å utføre en handling, for eksempel søke uten å fylt ut søkefelt, vil en bli varslet om at dette ikke er mulig og få en tilbakemelding på hva som må gjøres for at handlingen skal kunne fullføres.

Vi ønsket å designe et brukergrensesnitt som på best mulig måte opprettholder god og effektiv kommunikasjon mellom maskin og bruker. For å oppnå dette benyttet oss også av Ben Shneiderman *Eight Golden Rules of Interface Design* (Se 2.4.6) som omhandler strategier for å forbedre akkurat dette. Under utviklingen av brukergrensesnittet har vi forsøkt å følge de åtte reglene for design som er satt opp (Se 2.4.6). Men vi støtt på spesielt et problem rundt disse, knyttet til regel 3 *Tilby informativ tilbakemelding*. En bruker skal alltid få en informativ tilbakemelding ved utføring av handling, slik at en til en hver tid kjenner til statusen til systemet og at en handling fører til resultater. Basert på hvordan vi har implementert resultatsiden og søkefunksjonen vår på, er det ikke mulig å tilby bruker en tilbakemelding hvis en ønsker å skifte mellom sentimentanalyse og toneanalyse. Dette ser vi på som en svakhet. Dersom bruker er utålmodig kan vedkommende trykke på knappen flere ganger og starte flere operasjoner. Vedkommende kan også velge å gå tilbake til dashbordet og utføre søket på nytt med annet analysefilter.

Applikasjonen har blant annet blitt designet for å tilgjengeliggjøre sentimentanalyse. Brukergrensesnittet har blitt designet for på best mulig måte å vise fram statstikk og fokuset har ligget på å gjøre den brukervennlig for forskere. Det har derfor ikke vært stort fokus på universell utforming, ettersom applikasjonen primært skal bli brukt av en målgruppe. Til tross for dette har flere viktige retningslinjer for å oppnå universell utforming blitt fulgt. Aktiv bruk og valg av farger på ulike grafer for å skape assosiasjoner for bruker, så en enkelt og raskt kan skanne resultatsiden og se statstikken over søkeord eller frase. *Emotional tone* grafen består av linjer der hver har blitt tildelt en farge basert på hvilke følelse eller sentiment den respresenterer. Dette sees også på som en svakhet, og ikke universelt utformet, da en farge alene ikke kan informere om et resultat eller statstikk. Ikke alle assosierer en farge til tenkt resultat, samt flere lider av fargeblindhet. Donut grafene, som representerer analyseresultat av artiklene, ble derfor tildelt en *label*, slik at bruker også kan lese. *Emotional tone* grafen er derimot designet med fokus på en enkel visualisering for forskere, uten synlige *labels* på hver enkelt linje. De kan observeres hvis en holder mus over punkt på graf.

På dashbordet har fokuset ligget på å utvikle et brukergrensesnitt som er informativt, med et minimalistisk design. Dette har ført til enkelt problemer rundt universell utforming. Dersom bruker ønsker å legge på filter er boksene for å huke av nyhetskilder og følelser små, de ble designet for å ta minst mulig plass slik at all informasjon om hver enkelt boks fikk plass på en oversiktlig måte. Det samme gjelder knappen for å lukke filtervinduet. Denne ble designet for å være intuitiv i form av et kryss, uten fokus på størrelse på klikkflaten. Men som tidligere skrevet har fokuset ligget på å designe en enkel og intuitiv applikasjon som skal brukes av en målgruppe, ikke på universell utforming og tilpassing til flere brukergrupper.

5.3 Ingeniørfaglige resultater

I dette delkapittelet skal vi vurdere de ulike målene satt i visjonsdokumentet. Vi trekker frem eventuelle svakheter i målene satt og om dette kan påvirke måloppnåelse i drift. Måloppnåelse i prosjektet vises i kapitelet om resultater (Se kapp. 4.3).

1. *Systemet skal ha funksjonalitet for søking etter artikler og utføring av sentimentanalyse.*

   Problemstillingen fokuserer på å forenkle og tilgjengeliggjøre sentimentanalyse for forskere for å se trender i mediebildet, funksjonalitet tilknyttet dette kreves derfor av systemet. For å få utført sentimentanalyse på spesifikke ord eller fraser må applikasjonen tilby søke funksjonalitet. En bruker må kunne søke etter ord eller fraser, og søkefeltet må håndtere søkemotoregenskaper. I tillegg, for å tilgjengeliggjøre sentimentanalyse, skal sentimentanalysen utføres uten at det stilles noe annet krav til bruker enn søkeord og eventuelle filtre. Utføringen av analysen skal foregå på forhånd og lagres i databasen slik at søket ikke skal bli påvirket av tiden det tar å utføre en sentimentanalyse på en artikkel. En svakhet her er at en ikke kan analysere de nyeste artiklene skrevet i media, ettersom de først må *scrapes* av *webscraper* og så analyseres. Bruker må isåfall planlegge regelmessig *scraping*.

2. *Bruker skal kunne eksportere resultatet av sentimentanalysen til PDF.*

   Ettersom applikasjonen vår krever at bruker er koblet til Internett, vil det være viktig med en funksjon som lar bruker lagre resultat etter søk slik at en kan ta det med seg videre. Målgruppen av brukere er også tiltenkt forskere som vil kunne ønske å se på og presentere data, derfor viktig at dette vil være mulig uten internett tilkobling.

3. *Systemet skal ha en oppetid på 99.9%.*

   Applikasjonen er avhengig av IBM Watson sin NLP, samt databaseaksess i Cloud Foundry. Eventuelt nedetid der vil kunne påvirke målet om oppetid på 99.9%, og vil være en svakhet i systemt. Men applikasjon er for øyeblikket avhengig av disse ressursene og er bygd opp basert på de.

4. *Bruker trenger ikke opprette bruker eller lagre informasjon om seg selv for å få tilgang til systemet.*

   For å gjøre systemet så enkelt og raskt som mulig å ta i bruk, er det ikke nødvendig å kreve informasjon fra bruker eller kreve at de oppretter en profil. Applikasjonens hovedfokus er muligheten til å gjennomføre en sentimentanalyse på ønsket søkeord. Ettersom bruker ikke oppretter profil eller lagrer noe informasjon om seg selv som gjør det mulig å knytte bruker opp mot søk, vil det ikke være mulig å lagre søk. Dette kan bli sett på som en svakhet. Dersom bruker ønsker å gå tilbake til søk eller se på søk tidligere gjort med spesifikt søkeord og filter, vil en måtte utføre søk og legge på filtre på nytt.

5. *Systemet skal alltid være åpent og tilgjengelig.*

   Som nevnt tidligere er systemet avhengig av eksterne ressurser, IBM Cloud Foundry og IBM Watson, som kan påvirke om systemet er tilgjengelig. Dette er en svakhet i applikasjonen vår, men systemet er avhengig av disse ressursene og en del av oppdragsgiverens krav som bruk av ressurser til å få utført sentimentanalyse. Applikasjonen krever også internett for å kunne kjøres, som også kan bli sett på som en svakhet ettersom en ikke alltid har tilgang til dette. Men dette

er også noe systemet er avhengig av ettersom det er en web applikasjon som kjøres på nett og er avhengig av eksterne ressurser.

6. *Systemet krever lite vedlikehold.*

   Systemet er utviklet som en enkel web applikasjon som krever lite vedlikehold. Gruppen har fokusert på god kodekvalitet, slik at ved videre arbeid på applikasjonen skal det være enkelt å videreutvikle systemet og funksjonaliteten. Det skal være enkelt for nye utviklere å sette seg inn i koden.

7. *Det skal være mulig å integrasjonsteste applikasjonen i senere tid.*

   Alle tester og forsøk gjort under bachelorprosjektet skal være mulig å gjenskape av andre i senere tid. Dette har vi sørget for gjennom en grundig beskrivelse av hvordan testene er satt opp og gjennomført i kapittel knyttet til resultat, se 4.3.2. Grunnen til at dette har blitt prioritert er for at testene vi har gjort i løpet av bachelorprosjektet, skal kunne utføres av andre for å bevise at de stemmer og gir samme resultat.

8. *Applikasjonen skal ikke installeres på maskin eller mobile enheter, men vil være tilgjengelig på internett.*

   Ønske fra oppdragsgiver, når oppgaven ble lagd, var at det skulle lages en web applikasjon som skulle forenkle og tilgjengeliggjøre sentimentanalyse. Applikasjonen vår er per dags dato tilgjengelig over internett og krever ingen spesiell maskinvare eller mobil enhet for å kunne kjøres. Ved å lage en applikasjon tilgjengelig på Internett vil flere kunne benytte den.

9. *Det stilles ingen krav til bruker ved ny utrulling av applikasjon.*

   Ved ny utrulling av applikasjon vil applikasjonen ha en kort nedetid, i det applikasjonen bygges igjen. Det stilles ikke noen krav til bruker ved ny utrulling, med mindre det blir endring i design og funksjonalitet. Bruker må da eventuelt tilpasse seg det nye designet.

5.4 Evaluering av utviklingsmetodikk og arbeidsprosess

Gruppen valgte å benytte seg av ScrumBan som utviklingsprosess og brukte i forbindelse med dette Trello som produkttavle (Se 3.5). Vi skal først gå gjennom hva som fungerte bra og dårlig med denne prosessen, forså å se på hva vi ville ha gjort anerledes.

Hva fungerte bra ved bruk av ScrumBan

- Utviklnigsmetodikken ga oss en oversikt og struktur på måten vi arbeidet på. Trello ga oss muligheten til å se hva andre jobbet med, samt hva som måtte gjøres videre.
- Mulighetene til å legge til oppgaver underveis i en sprint og oppdatere *Backlog*. Ettersom vi som utviklere har en tendens til å komme på ny tilleggsfunksjonalitet underveis i utviklingen, satte vi stor pris på at det var mulig å sette opp dette som oppgaver underveis.

Hva fungerte dårlig ved bruk av ScrumBan

- Å ta i bruk Trello og sette opp oppgavene i Trello i starten av prosessen ble ikke fulgt opp bra

nok. Det ble satt opp noen for store oppgaver, slik at tavlen ikke ble oppdatert, endret seg, over enkelte dager.

- Mot slutten av prosjektet var det bare en som jobbet på koden med å fikse *bugs*. Dette skal ikke føres opp på Trello ettersom det er problemer man oppdager i mens man tester koden. Trello tavlen kunne derfor ikke benyttes i siste del, mot slutten av prosjektet.

Å bruke ScrumBan som utviklingsmetodikk under utviklingen av dette systemet hadde både positive og negative sider, som nevnt over. Det ga oss en god oversikt over hva som måtte gjøres av oppgaver, hvilke som var under utvikling og hvilke som var fullført. Ettersom vi hadde et prosjekt som både bestod av en systemutviklingsdel og en maskinlæringsdel var det noe kronglete å sette sammen et Gantt Diagram og sprinter til ScrumBam. Dette resulterte i at vi lagde et Gantt diagram der hvert enkelt medlem i gruppen satte opp sine egne sprinter, både innenfor systemutvikling og maskinlæring. Gantt diagrammet ga oss en oversikt over hele bachelorprosjektet og ikke bare de som gikk under utvikling av system. Dette måtte så tilpasses ScrumBan, ettersom en sprint bare baserte seg på oppgaver innenfor et område; *Backend*, *Frontend* eller *maskinlæring*. Som skrevet i resultater, ble sprintene oppført i Gantt diagrammet slått sammen, slik at en sprint i Trello både inneholdt *Frontend* og *Backend*. Dette innebar også at enkelte sprinter måtte gå over flere sprinter i ScrumBan metodikken, ettersom de bestod av større oppgaver som tok lengre tid å fullføre enn andre. Dette var det vi så på som mest negativt ved å benytte oss av utviklingsmetodikk, som ScrumBan, på et bachelorprosjekt som består av mer enn bare en systemutviklingdel. Gruppen har innsett at det mest optimale, under utførelsen av et bachelorprosjekt som vårt, hadde vært en metodikk mer egnet for forskningsprosjekter. En mulig løsning ville ha vært å kombinere en forskningmetodikk og en utviklingsmetodikk, eller funnet et metodikk som tar hensyn til begge. Men ettersom dette var første gang gruppen har utført et slikt prosjekt, samt at vi kun var tre medlemmer, var ikke dette noe vi vurderte ved start av prosjekt.

Som en kan se i Gantt diagrammet er det ikke skrevet flere sprinter for *Backend* etter 22.03.19, tiltross for at vi har jobbet med og satt opp nye mål i ScrumBan etter denne datoen. Dette er fordi all planlagt funksjonalitet, skrevet som funksjonelle egenskaper i visjonsdokumentet, skulle være ferdig til 22.03.19. Og som en kan se ved å studere Gantt diagrammet og statusoppdateringene, fullførte vi dette innen gitt dato. Oppgavene ført opp i Trello etter denne datoen, som omhandler *Backend*, er tilleggsfunksjonalitet vi har valgt å legge til senere. Som tidligere nevnt, ønsket vi en utviklingsmetodikk som tillot oss å legge til oppgaver senere i prosessen ettersom vi var relativt sikre på at kom til å ville legge på tilleggsfunksjonalitet etterhvert i systemutviklingen. Arbeid knyttet til administrasjonssiden og *Webscraper*verktøyet er eksempler på tilleggsfunksjonalitet vi utviklet etter 22.03.19.

Et annet problem som oppstod under utvikling av systemet, var at tiltross for at vi ble ferdig med en oppgave satt opp i en sprint til riktig tid, måtte vi senere i utvikling tilbake å endre funksjonalitet knyttet til denne. For eksempel under utviklingen av WebScraper APIet. Det ble her laget et API, og tilhørende funksjonalitet ble lagt til, innenfor tidsintervallet satt opp, men et par uker senere ble *Frontenden* endret på som gjorde at nyhetsartiklene måtte hentes inn annerledes. Som en konsekvens av dette, måtte WebScraper APIet endres på og en dag ble brukt til dette. Til tross for at problemer som dette har oppstått underveis, har utviklingsmetodikken fungert ypperlig for vårt prosjekt ettersom vi kunne legge til oppgaver underveis. Gantt diagrammet har derimot ikke blitt endret.

5.5 Helhetlig systemperspektiv

Systemet som har blitt utviklet, under utførelsen av dette bachelorprosjektet, er en enkel og intuitiv web applikasjon. Det stiller ingen krav til bruker om maskinvare eller lignende, kun tilkobling til internett og tilgang til en nettleser. Dette gjør at blant annet forskere kan ta systemet lett i bruk. Hensikten har vært å forenkle og tilgjengeliggjøre sentimentanalyse av trender i mediebildet for forskere, og produktet tilbyr denne funksjonaliteten per dags dato. Dette vil være et godt verktøy for forskere som studerer og ønsker å se hvordan ulike trender i mediebildet kan påvirke resultater knyttet til de trendene (Se tidligere arbeid 2.1.3). Systemet vil gjøre det enklere for de å se sammenhenger mellom resultat og trend i mediebildet. Samtidig vil det gjøre det mer økonomisk ettersom dette er en gratis tjeneste tilgjengelig på internett. En trenger heller ikke utføre noe manuelt arbeid, som å lese gjennom og studere mediebildet selv, som også gjør det mer økonomisk å gjennomføre en slik analyse.

En mulig etiskproblemstilling som kan oppstå ved bruk av dette produktet er knyttet falske nyheter. Vi vet at vår WS har et begrenset utvalg nyhetskilder som støttes, noe som kan gi et feilaktig mediebildet. Et eksempel på dette kan være nyhetskilder som skriver usant eller personlig, som kan gi fremtidig bruker av systemet unøyaktig informasjon om disse temaene. En kan dra eksempelet til virkeligheten, hvor det går anklagelser om valgjuks i USA. Vårt system kan da bli brukt som et verktøy for å påvirke velgere i en retning slik som Cambridge Analytica og andre selskaper som er beskyldt for å være medvirkende i dette.[84]

5.6 Gruppearbeidet

Ettersom gruppen har bestått av medlemmer med ulike ferdigheter og tanker om hva de ønsker å lære mer om, har det vært ypperlig å arbeide sammen og fordele arbeidet på den måten vi har gjort. Hvert enkelt medlem har fått muligheten til å utvikle sitt ferdighetsnivå innenfor det området vedkommende ønsker å studere videre. Det har også vært god kommunikasjon innad i gruppen, og vi har hjulpet hverandre på tvers av arbeidsoppgaver. Ettersom vi har jobbet sammen tidligere har vi vært klar over hverandres styrkene og hvordan vi på best mulig måte kan arbeide sammen. Vi har ikke støtt på noen problemer når det kommer til uenigheter eller lignende. Dersom medlemmer har vært uenige om en sak, har vi diskutert denne innad i gruppen og kommet fram til best mulig løsning. Ingen har satt seg fast i noen diskusjoner og alle har vært åpne for å komme fram til en felles enighet. Alt i alt har dette vært en svært lærerik periode, der alle har lært mye om seg selv og hva en ønsker å lære mere om i framtiden.

6 Konklusjon

Problemstillingene fra toppen av rapporten er som følger:

1. Hvordan utføres sentimentanalyse på norsk og hvilke utfordringer kan dette by på?
    1. Hvilke utfordringer har kjente annoterte norske datasett for sentimentanalyse?
    2. Hvordan presterer ulike modeller på norsk sentimentanalyse?
2. Utvikling av en applikasjon som forenkler og tilgjengeliggjør sentimentanalyse for forskere som ønsker å studere trender i mediebildet.

Den største utfordringen med norsk sentimentanalyse er at det finnes få kjente annoterte norske datasett for oppgaven, og av den grunn lagde UiO datasettet NoReC. Ved å trene en modell i fastText med NoReC datasettet fikk vi nøyaktigheten 72,0%. Utfordringen med NoReC er at det består av få, men lange eksempler. Samtidig er eksemplene så store at et terningkast blir for generelt til å annotere hele teksten. Videre finnes det heller ikke nøytrale terningkast. En løsning på utfordringen er å lage et eget datasett ved bruk av Twitter API. En svakhet ved Twitter APIet er at vi vil få tweets som ikke er norske til tross for at vi ber om kun norske tweets. Dette ble bedre ved sjekke hver tweet opp mot norske ordlister. Å benytte seg av stemming gjorde ikke noen forskjell for datasettet, og vi fikk nøyaktigheten 81,6% ved å modellere i fastText. Ved å mikse datasettene tilgjengelig vil nøyaktigheten bli 77,7%.

Ulike modeller som kan brukes til sentimentanalyse er fastText og LSTM, som fikk henholdsvis 81,6% og 78,4% etter trening med det norske ustemte datasettet av tweets. FastText brukte 45 sekunder på treningen med en Macbook Pro med 2 kjerner, mens LSTM brukte 6 min på en virtuell maskin med 32 kjerner. Det ble ikke brukt like mye tid på å tilpasse hyperparametrene til LSTM, og den trenger ikke like stort datasett for å få god presisjon slik som fastText gjør. Dersom man har et stort datasett, og ikke ønsker å bruke mye tid, er det i følge disse resultatene bedre å bruke fastText.

Etter bruker har fått gjennomført en sentimentanalyse, vil det være ønskelig å få vist fram resultatene på en oversiktlig måte. Systemet forenkler og tilgjengeliggjør sentimentanalyse på enkelte nyhetskilder, og presenterer hva vi mener er viktig statistikk, ved hjelp av informasjonsvisualisering. For fremtidige brukere kan det være nødvendig å ha muligheten til å få analysert nyhetskilder systemet ikke er satt opp med. Det er derfor lagt til funksjonalitet for at administratorer skal kunne legge til *hosts*, endre disse og planlegge henting av artikler fra nyhetskildene. Systemet er tilpasset brukere som ønsker å studere mediebildet der en vil ha innsikt i gjennomsnittet av store mengder data.

6.1 Videre arbeid

Systemet har mange funksjonelle egenskaper satt opp fra start, men det har også blitt lagt til flere i løpet av utviklingsprosessen. Her er hva vi gjerne skulle ha implementert, men ikke fikk tid til.

Produktet skulle gjerne hatt brukerprofiler hvor tidligere søk blir lagret, samt er lett tilgjengelige. Videre skulle vi ha implementert støtte for personlige kilder til indeksering. Dette kan være beskyttede nettsteder hvor man trenger en API nøkkel eller liknende for å få tilgang. Dette kan også være bedriften/institusjonens egne ressurser som omverden ikke skal ha tilgang til. Det ville også vært ønskelig at produktet kan kjøres på en lokal maskin, hvor informasjon kan hentes og analyseres direkte fra filsystemet. Eksempelvis, en forsker som har data lagret på egen maskin som ikke kan være tilgjengelig over internett. For denne type funksjonalitet ville det vært ønskelig å presentere applikasjonen som et dataprogram som kjøres lokalt, ikke som en webapplikasjon.

Av allerede eksisterende funksjonalitet som vi gjerne skulle utvidet har vi PDF eksporteringen på resultatsiden. Vi ville endret funskjonaliteten fra å benytte seg av utskriftsvinduet til operativsystemet, til å generere en PDF fil som lastes ned til maskinen. Mot slutten ble administratorsiden lagt til produktet, og da dette var ekstra funksjonalitet ble ikke utseende på denne høyt prioritert. Endring av designet på denne siden vil være noe vi ønsker å gjøre i nærmeste framtid. Videre ønsker vi at *web scraper*-en sitt reportoar av nyhetskilder skal utvides. Alt for å gi et mer helhetlig mediabilde, samt for å øke indeksen over søkbare ord og fraser.

I systemdokumentasjonen nevnes det noen utfordringer med Travis CI som kontinuerlig integrasjonsverktøy. På bakgrunn av dette ønsker vi å få integrasjonen med TravisCI på plass for å forenkle og sikre videreutvikling av systemet.

I maskinlæringen ønsker vi å lete videre etter de beste hyperparameterne for modellen vår. Vi ønsker også å finne eller lage et bedre norsk annotert datasett. Siden LSTM vil fungere godt på små datasett er ikke størrelse hva vi er på jakt etter, men bedre annotering og setningene er bedre skrevet. Til slutt ønsker vi å se på en LSTM modell med flere lag, for å øke presisjonen til modellen.

Produktet baserer seg på IBM Watson sin NLU for sentimentanalyse. Vi ønsker å trene vår egen modell, både til sentimentanalyse på entiteter i kontekst og en modell for entitetsanalyse, for å finne sentiment på entiteter i teksten. Dette for å koble vår egen modell opp mot produktet for å gi et bredere utvalg av analyser brukeren kan benytte seg av og støtte for norske nyhetskilder.

Bibliografi

[1] John Arne Øye. Sentiment analysis of norwegian twitter messages. 2015.

[2] Patrik Fridberg Bakken and Terje Bratlie. Sentiment analysis in norwegian political news - employing replicated methods and experimental features to understand a complex domain. 2016.

[3] Venkata Sasank Pagolu, Kamal Nayan Reddy Challa, Ganapati Panda, and Babita Majhi. Sentiment analysis of twitter data for predicting stock market movements. 28 Oct 2016.

[4] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[5] Anjali Jivani. A comparative study of stemming algorithms. *Int. J. Tech. Appl.*, 2:1930–1938, 11 2011.

[6] What is Machine Learning. https://emerj.com/ai-glossary-terms/what-is-machine-learning. (Besøkt April. 2019).

[7] Machine learning crash course. https://developers.google.com/machine-learning/crash-course/, 05. Mars 2019. (Besøkt April. 2019).

[8] Jason Rowe. Tre ting du må vite om kunstig intelligens. https://www2.deloitte.com/no/no/pages/technology/articles/tre-ting-vite-kunstig-intelligens-ai.html. (Besøkt Mai. 2019).

[9] Shubhadeep Mukherjee and Pradip Kumar Bala. Detecting sarcasm in customer tweets: an nlp based approach. *Industrial Management & Data Systems*, 117(6):1109–1126, 2017.

[10] Text classification: A comprehensive guide to classifying text with machine learning. https://monkeylearn.com/text-classification/. (Besøkt Mai. 2019).

[11] 5 things you need to know about sentiment analysis and classification. https://www.kdnuggets.com/2018/03/5-things-sentiment-analysis-classification.html.

[12] Hidden layer definition. https://www.techopedia.com/definition/33264/hidden-layer-neural-networks. (Besøkt Mai 2019).

[13] Neural network models in r. https://www.datacamp.com/community/tutorials/neural-network-models-r, 22.01.2019. (Besøkt April. 2019).

[14] When to use mlp, cnn, and rnn neural networks. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/. (Besøkt Apr. 2019).

[15] Margaret Rouse and Ed Burns. Deep neural network. https://searchenterpriseai.techtarget.com/definition/deep-learning-deep-neural-network, January 2018. (Besøkt April. 2019).

[16] How to avoid overfitting in deep learning. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. (Besøkt Apr. 2019).

[17] How to split your dataset to train and test datasets using scikit learn. https://medium.com/@contactsunny/how-to-split-your-dataset-to-train-and-test-datasets-using-scikit-learn-e7cf6eb5e0d. (Besøkt Apr. 2019).

[18] Machine learning(ml) - data preprocessing. https://medium.com/datadriveninvestor/machine-learning-ml-data-preprocessing-5b346766fc48. (Besøkt Mars 2019).

[19] David A. Hull. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84.

[20] Stemming and lemmatization. https://queryunderstanding.com/stemming-and-lemmatization-6c086742fe45. (Besøkt Mai. 2019).

[21] Natural language processing. https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/20_Natural_Language_Processing.ipynb. (Besøkt Apr. 2019).

[22] Word embedding sentiment classification using keras. https://towardsdatascience.com/machine-learning-word-embedding-sentiment-classification-using-keras-b83c28087456. (Besøkt Mars. 2019).

[23] Activation functions in neural networks. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. (Besøkt Apr. 2019).

[24] Adam optimization. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. (Besøkt Apr. 2019).

[25] Accuracy, precision, recall or f1? https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. (Besøkt Mars 2019).

[26] Virtuell maskin. https://it.uib.no/Virtuell_maskin. (Besøkt Mai 2019).

[27] Kontinuerlig integrasjon. https://ntnu.blackboard.com/courses/1/194_TDAT2003_A_2017_H_1/content/_98791_1/embedded/CI-foiler.pdf, 2015/2016. (Besøkt April. 2019).

[28] Digital samhandling: Tre gode grunner til hvorfor det gir deg konkurransefortrinn. https://blog.soprasteria.no/blog/2017/11/07/digital-samhandling-tre-gode-grunner-til-hvorfor-det-gir-deg-konkurransefortrinn/. (Besøkt Mai 2019).

[29] Komme i gang - om versjonskontroll. https://git-scm.com/book/no-nb/v1/Komme-i-gang-Om-versjonskontroll, Note = (Besøkt April. 2019),.

[30] Skytjenester. https://www.datatilsynet.no/personvern-pa-ulike-omrader/internett-og-apper/skytjenester/. (Besøkt April. 2019).

[31] Yngve Dahl. Konseptuelle- og mentale modeller. http://folk.ntnu.no/baldurk/skolearbeid/MMI/Forelesninger%20MMI/41-Konseptuelle-%20og%20mentale%20modeller.pdf, vår 2017. (Besøkt Mai. 2019).

[32] Introduksjon til interaksjonsdesign. https://ntnu.blackboard.com/bbcswebdav/pid-98794-dt-content-rid-1433468_1/courses/194_TDAT2003_A_2017_H_1/Leksjon_%20Intro%20til%20interaksjonsdesign.pdf, September 2016. (Besøkt April. 2019).

[33] D. Norman. *The Design of Everyday Things*. Basic Books, 1988.

[34] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., 1986.

[35] Introduksjon til interaksjonsdesign. https://uu.difi.no/kva-er-universell-utforming. (Besøkt April. 2019).

[36] W.V. Siricharoen. Infographics: The new communication tools in digital age. https://www.researchgate.net/profile/Waralak_Siricharoen/publication/256504130_Infographics_the_new_communication_tools_in_digital_age/links/0c9605232e6f666b1f000000.pdf, 6.January, 2019. (Besøkt Jan. 2019).

[37] S. Card J.D. Mackinlay, B. Shneiderman. *Readings in Information Visualization: Using Vision to Think (Interactive Technologies)*. Morgan Kaufmann Publishers Inc., 1999.

[38] T. Tullis S. Djamasbi, M. Siegel. Visual hierarchy and viewing behavior: An eye tracking study. https://digitalcommons.wpi.edu/cgi/viewcontent.cgi?article=1018&context=uxdmrl-pubs, 2011. (Besøkt Jan. 2019).

[39] M. Soegaard. Visual hierarchy: Organizing content to follow natural eye movement patterns). https://www.interaction-design.org/literature/article/visual-hierarchy-organizing-content-to-follow-natural-eye-movement-patterns, 8.Mai, 2019. (Besøkt Jan. 2019).

[40] S. Mialki. The z-pattern layout: What it is, why it works, and when to use it. https://instapage.com/blog/z-pattern-layout, 10.September, 2018. (Besøkt Jan. 2019).

[41] Ibm watson nlu. https://cloud.ibm.com/apidocs/natural-language-understanding?code=python. (Besøkt Februar 2019).

[42] Adam Bittlingmayer. Amazon reviews for sentiment analysis. https://www.kaggle.com/bittlingmayer/amazonreviews. (Besøkt Mars. 2019).

[43] Introduction to word embedding and word2vec. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa. (Besøkt Apr. 2019).

[44] Fasttext text classifier tutorial. https://fasttext.cc/docs/en/supervised-tutorial.html. (Besøkt Jan. 2019).

[45] Fasttext under the hood. https://towardsdatascience.com/fasttext-under-the-hood-11efc57b2b3. (Besøkt Mars 2019).

[46] Imdb dataset. https://www.kaggle.com/c/word2vec-nlp-tutorial/data. (Besøkt Apr. 2019).

[47] Erik Velldal, Lilja Øvrelid, Eivind Alexander Bergem, Cathrine Stadsnes, Samia Touileb, and Fredrik Jørgensen. NoReC: The norwegian review corpus. Common Language Resources and Technology Infrastructure Norway (CLARINO) Bergen Repository.

[48] Package for a simple interface for working with the norec dataset. https://github.com/ltgoslo/norec/tree/master/src.

[49] Hva betyr et terningkast? https://p3.no/filmpolitiet/2011/07/hva-betyr-et-terningkast/.

[50] Middelmådig. https://www.naob.no/ordbok/middelmådig. (Besøkt Mai 2019).

[51] Norwegian stemming algorithm. https://snowballstem.org/algorithms/norwegian/stemmer.html. (Besøkt Mai. 2019).

[52] Norske ordlister. https://github.com/0301/ordliste. (Besøkt Apr. 2019).

[53] Afinn ordliste. https://github.com/olavski/afinn. (Besøkt Apr. 2019).

[54] Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. Emoji sentiment ranking 1.0, 2015. Slovenian language resource repository CLARIN.SI. (Besøkt Apr. 2019).

[55] Yanqing Chen and Steven Skiena. Building sentiment lexicons for all major languages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 383–389, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[56] How to load machine learning data in python. https://machinelearningmastery.com/load-machine-learning-data-python/. (Besøkt Apr. 2019).

[57] Pandas - python data analysis library. https://pandas.pydata.org. (Besøkt Apr. 2019).

[58] Scikit-learn. https://scikit-learn.org/stable. (Besøkt Apr. 2019).

[59] Tensorflow. https://www.tensorflow.org/guide/keras. (Besøkt Apr. 2019).

[60] Tensorboard. https://www.tensorflow.org/guide/summaries_and_tensorboard. (Besøkt Mai 2019).

[61] Jupyter notebook. https://jupyter.org. (Besøkt Apr. 2019).

[62] Tensorflow tutorial 20 natural language processing. https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/20_Natural_Language_Processing.ipynb.

[63] Ou Wu, Tao Yang, Mengyang Li, and Ming Li. ρ-hot lexicon embedding-based two-level LSTM for sentiment analysis. *CoRR*, abs/1803.07771, 2018.

[64] L. Bradford. What is github and why should i use it? https://www.thebalancecareers.com/what-is-github-and-why-should-i-use-it-2071946, 6.January, 2019.

[65] Travis ci. https://travis-ci.org/. (Besøkt Apr. 2019).

[66] Continuous integration. https://github.com/marketplace/category/continuous-integration. (Besøkt Apr. 2019).

[67] Natural language understanding. https://cloud.ibm.com/apidocs/natural-language-understanding#text-analytics-features. (Besøkt Jan. 2019).

[68] Cloudant. https://cloud.ibm.com/catalog/services/cloudant. (Besøkt Jan. 2019).

[69] Db2. https://cloud.ibm.com/catalog/services/db2. (Besøkt Mai. 2019).

[70] Cloud foundry. https://cloud.ibm.com/cloudfoundry/overview. (Besøkt Mai. 2019).

[71] React. https://radar.bekk.no/tech2018/sprak-og-rammeverk/react. (Besøkt Mai. 2019).

[72] Context. https://reactjs.org/docs/context.html. (Besøkt April. 2019).

[73] About node.js. https://nodejs.org/en/about/. (Besøkt April. 2019).

[74] Jest 24.6. https://jestjs.io/. (Besøkt Feb. 2019).

[75] What is babel? https://babeljs.io/docs/en/index.html. (Besøkt Apr. 2019).

[76] Google news api. https://newsapi.org/s/google-news-api. (Besøkt Feb. 2019).

[77] Auth0 overview. https://auth0.com/docs/getting-started/overview. (Besøkt Apr. 2019).

[78] Live with perspective. https://design.trello.com/perspective. (Besøkt Mai. 2019).

[79] J. Nielsen. F-shaped pattern for reading web content (original study)). https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content-discovered/, 17.April, 2006. (Besøkt Jan. 2019).

[80] Prosjekthåndbok. Se vedlagt zip-fil. (Besøkt Mai 2019).

[81] Eylean. *The Ultimate Agile Guide*. eBook. (Besøkt Jan. 2019).

[82] Visjonsdokument. Se vedlagt zip-fil. (Besøkt Mai 2019).

[83] Kildekode. Se vedlagt zip-fil. (Besøkt Mai 2019).

[84] Facebook-cambridge analytica data scandal. https://en.wikipedia.org/wiki/Facebook\T1\textendashCambridge_Analytica_data_scandal#Use_of_the_data. (Besøkt Mai 2019).

NTNU
Kunnskap for en bedre verden

IBM