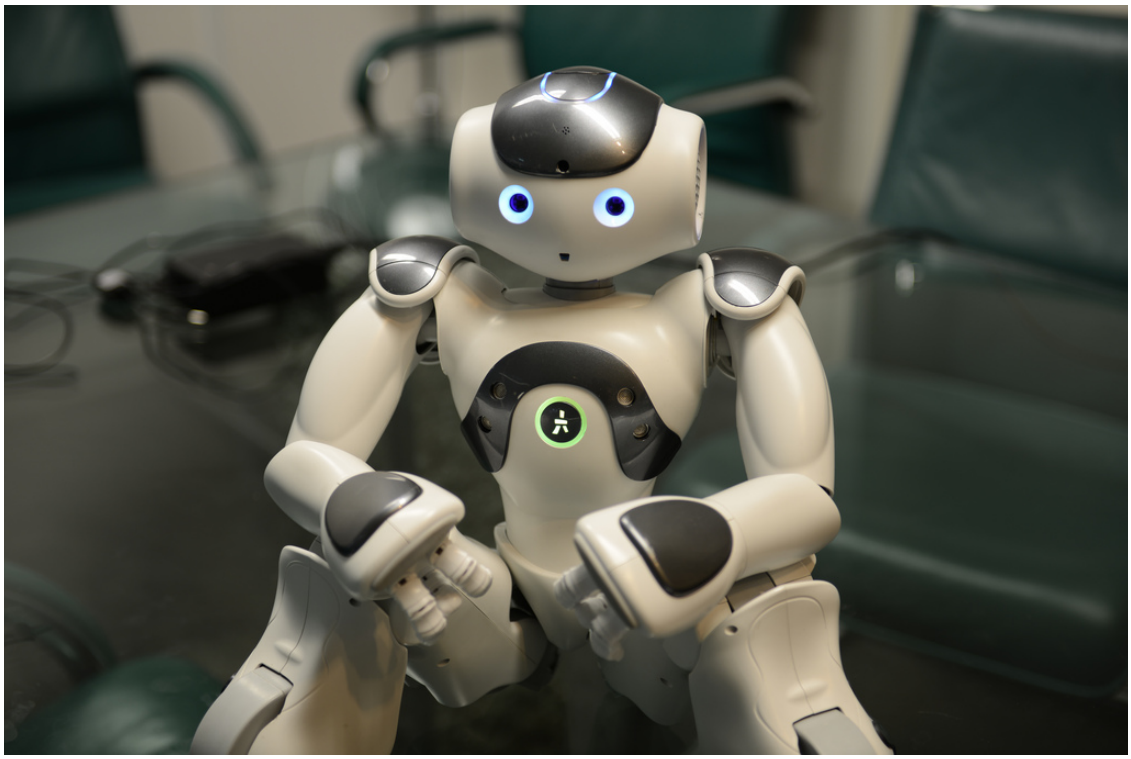
CHALMERS



ChatUp

En chatbot med kontinuerligt lyssnande och snabb respons

Examensarbete i Data och Informationsteknik

JOHAN MARTINSON

DANIEL RYDÈN NAKHLE

Instutitionen för Data och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2018

EXAMENSARBETE 2018:16

ChatUp

En chatbot med kontinuerligt lyssnande och snabb respons

JOHAN MARTINSON
DANIEL RYDÈN NAKHLE

CHALMERS

Instutitionen för Data och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg, Sverige 2018

ChatUp
En chatbot med kontinuerligt lyssnande och snabb respons
JOHAN MARTINSON
DANIEL RYDÈN NAKHLE

Examiner: Peter Lundin, Instutitionen för Data och Informationsteknik

Instutitionen för Data och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
SE-412 96 Göteborg
Sverige Telephone +46 (0)31 772 1000

Omslag: Robotmodellen som mjukvaran har installerats i.

Instutitionen för Data och Informationsteknik
Göteborg, Sverige 2018

ChatUp
En chatbot med kontinuerligt lyssnande och snabb respons
DANIEL RYDÈN NAKHLE
JOHAN MARTINSON
Instutitionen för Data och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet

Examensarbete

Sammanfattning

Intresset för IT och framförallt artificiell intelligens ökar i svenska företag för varje dag som går. Därför har en plattform i form av en robot utvecklats för att tillgodose experiment inom maskininlärning och maskinintelligens. Tanken är att denna basprodukt skall fånga människors intresse under mässor och event. Den utvecklade mjukvarumodulen ChatUp har integrerats på en Nao-robot, som är en liten människoliknande robot, som tar in användares förfrågningar, via en NLP tolkar förfrågan och sedan väljer en lämplig tjänst för att slutligen svara. Då tanken är att roboten ska uppträda så människolikt som möjligt ska tiden mellan förfrågan och svar förkortas till ett minimum. Projektet har därför undersökt ljudsegmentationsgsalgoritmer, för att ta bort tystnad och icke-tal i ljud detta med avsikt att minimera storleken på dess datapaket som skickas över nätet och på så sätt förkorta responstiden. Undersökningen har resulterat i vetskap om att manipulation av live-strömmat ljud inte lämpar sig medan analys vid live-strömmning kan förkorta responstiden märkbart. Resultaten ska användas som grund för fortsatt utveckling av roboten och ChatUp modulen.

Nyckelord: NLP, ljudsegmentationsalgoritm, responstid, chatbot, artificiell intelligens, zero crossing rate, short term energy, live-strömmning

Abstract

Interest in IT and especially artificial intelligence increases in Swedish companies by the day. Therefore a platform in the form of a robot to accommodate experiments in machine learning and machine intelligence has been developed. The idea is that this basic product will capture peoples' interests during fairs and events. The developed software module ChatUp has been integrated into a Nao-robot, which is a small humalike robot, that takes in user requests, interprets them via a NLP and then selects an appropriate service to finally respond. Since the idea is that the robot should behave as humanlike as possible, the time between request and response should be shortened to a minimum. The project has therefore investigated sound segmentation algorithms, to remove silence and non-speech in audio this with the intention of minimizing the size of its data packet transmitted over the network, thus shortening the response time. The investigation has resulted in the fact that manipulation of live-streaming audio is not suitable while live streaming analysis can shorten response time noticeably. The results will be used as a basis for further development of the robot and the ChatUp module.

Keywords: NLP, audio segmentation algorithm, response time, chatbot, artificial intelligence, zero crossing rate, short term energy, live-streaming

Förord

ChatUp är ett examensarbete på kandidatnivå som utförts av två studenter på högskoleingenjörsprogrammet i datateknik. Examensarbetets omfattning motsvarar en halv termins heltidsstudier vid institutionen för Data- och informationsteknik på Chalmers tekniska högskola i Göteborg. ChatUp skedde i samarbete med Cybercom Group i Göteborg.

Vi vill framföra ett stort tack till Philip Laine, från Cybercom, som trots begränsad tid och uppdrag på annan plats varit en utmärkt handledare under examensarbetets gång. Vi vill även tacka Gabriel Ibanez, ledare för Cybercom Innovation Zone i Göteborg, som gav oss möjlighet att skriva examensarbete hos Cybercom. Sist men inte minst vill vi framföra ett stort tack till Sakib Sistek, vår handledare på Chalmers.

Johan Martinson, Daniel Rydén Nahkle

Göteborg, Juni 2018

Innehåll

Beteckningar

IT - Informationsteknik

MVP - Minimum Viable Product, minsta möjliga produkt. en version med tillräcklig funktionalitet för att uppdragsgivarna ska vara nöjda.

TTS - Text To Speech, en funktion som skapar röstliknande ljud ifrån en textsträng.

API - Aplication Programing Interface, är en uppsättning funktionaliteter som gör det möjligt för en applikation att kommunicera med en bakomliggande tjänst.

Chatbot - Det är ett datorprogram som håller konversationer via ljud eller text.

VAD - Voice Activity Detection är en algoritm som analyserar ljud efter röstinnehåll.

MATLAB - Matlab är ett programspråk som ofta används till matematiska och tekniska beräkningar så som vid signalbehandling.

Fallback - Fallback är en reservutväg för en chattbot att ta när den inte har ett specifikt svar på en fråga.

1
Inledning

I detta kapitel beskrivs bakgrunden till projektet. Därefter följer en beskrivning av dess syfte, samt en mer detaljerad redogörelse för vad som skall uppnås. Slutligen fastställs projektets avgränsningar.

1.1 Bakgrund

Intresset för data och informationsteknik ökar trots det behöver företag i Sverige och resten av världen fler arbetare inom data och informationsteknik. Under de senaste åren har det diskuterats i media och bland företag hur man får människor intresserade av IT. Flertalet lösningar har föreslagits så som programmering i Grundskolan eller karriärsajter och kampanjer för ungdomar [1], [2].

För att sätta sin prägel på debatten har Cybercom Group, i samarbete med Ericsson, börjat utveckla en hjärna hos en Nao-robot, vilket är en liten människoliknande robot. Detta i syfte att öka intresset för data och informationsteknik. Genom att utveckla roboten med aktuell teknik som maskininlärning och maskinintelligens vill man visa möjligheterna som finns med IT.

Förra året genomfördes ett projektarbete som tog fram en basfunktionalitet hos roboten. I nuläget tar det lång tid för roboten mellan förfrågning och respons; därför vill man utveckla roboten med kontinuerligt lyssnande samt förbättra den nuvarande svarsmekanismen. Utöver det vill man skapa möjligheter för roboten att initiera konversationer självmant samt att efterlikna mänskliga konversationer.

1.2 Syfte

Projektet syftar till utveckling av en mjukvarumodul, ChatUp, till en social robot som lyssnar på vad som sägs och bedömer ifall frågan är riktad till roboten. Man vill att flödet från att en fråga ställs till att svars ges ska gå snabbare än i nuläget. Ur ett användarperspektiv vill man göra det sömlöst att interagera med den sociala roboten, med andra ord vill man skapa en så människolik konversation som möjligt.

1.3 Mål

Den sociala roboten ska smidigt kunna förstå och svara på frågor. Krav på produkten är följande:

- Roboten ska kontinuerligt lyssna efter förfrågningar.
- Om frågan ställs till någon annan än roboten ska den ignorera förfrågan.
- Responstiden skall sänkas till under 5 sekunder då förfrågan är under 5 sekunder
- Roboten ska kunna svara på frågor inom sex specifika kategorier och när en fråga inte faller inom dessa ge någon form utav respons som är relevant till frågan.

  Kategorierna är följande:

  1. Användaren vill veta vad det är för lunchmeny i antingen kafeterian eller L's Kitchen.
  2. Användaren vill att roboten ska dansa.
  3. Användaren vill att roboten ska berätta ett skämt.
  4. Användaren vill veta när nästa buss avgår.
  5. Användaren vill veta var i byggnaden ett specifikt rum ligger.
  6. Användaren vill veta om en viss person är i möte eller på sitt kontor.

1.4 Avgränsningar

Avgränsningar för ChatUp är:

- Projektet ska baseras på Python.
- Eftersom hårdvaran redan finns så kommer endast mjukvara att utvecklas.
- Förmågan att ha konversationer utanför de sex kategorier behöver inte vara avancerad utan endast någon form av respons som är relaterat till den indata roboten får ska ges tillbaka.
- Roboten kommer inte självmant initiera konversationer utan användaren måste initiera konversationen med roboten.
- Huvudfokus kommer att ligga på att utveckla kontinuerligt lyssnande.
- Vi kommer enbart att försöka förbättra responstiden med hjälp av ljudsegemntations- och taligenkännings-analysalgoritmer.

2
Teknisk Bakgrund

I följande kapitel beskrivs vilka verktyg som använts i projektet. Även teorin bakom vissa nyckelkoncept och standarder, som behövs för att förstå projektet, beskrivs.

2.1 Python

Python är ett objektorienterat högnivåspråk som är designat för att vara enkelt att läsa då formatering är inbyggt i språket. Python har ett stort standardbibliotek som förser användare med verktyg som är anpassade för flertalet uppgifter. I Python har man även förmågan att skapa virtuella miljöer där man kan ha projekt specifika versioner av bibliotek. Detta är bra om man använder hårdvara som inte är så kraftfull eller andra bibliotek inte har kompatibilitet till den nyaste versionen [3].

2.2 PyAudio

PyAudio är ett Python-bibliotek som ger tillgång till I/O (Input/Output) funktioner för ljud. PyAudio har ett brett utbud av olika funktioner möjligheten att välja vilken enhet som ska användas för inspelning, format på ljudet, frekvens och om man spelar in i mono eller stereo. I detta projekt används det för att hantera strömning utav ljud [4].

2.3 Natural Language Processing (NLP)

Natural Language Processing är en gren utav artificiel intelligens som är talanalys i syfte för att få datorer att uppnå människolik uppfattning av språk. NLP drar från många discipliner så som datavetenskap och datalingvistik. Uppgiften är att bryta ner språk och meningar till små delar och sedan förstå relationerna mellan delarna. Det finns flertalet olika företag som erbjuder en NLP-tjänst [5].

Dialogflow

Dialogflow är en tjänst som erbjuder NLP och lär sig tolka meningar och fraser med hjälp av avsikter och entiteter. Dialogflow har stöd för strömning av ljud och ljudfiler och har förmågan att spara kontext från en tidigare session. Dialogflow stödjer inte så många språk och är komplex att använda vilket kräver en del tid för att komma igång med ordentligt [6].

2.4 Cleverbot

Cleverbot är en tredjepartstjänst som specialiserar sig på småprat och förmågan att ha konversationer om vad som helst. Cleverbot klarar pågrund av träning sedan den artificiella intelligensen skapades 1988 av detta. När hemsidan skapades 2006 började Cleverbot tränas på en global skala. Företaget erbjuder folk att skicka förfrågningar till deras *Application Programming Interface* (API) vilket är det som används i projektet [7].

2.5 JSON-objekt

JavaScript Object Notation (JSON), är ett fil-format för att skicka data mellan webtjänster. Detta format är enkelt att läsa och att använda på grund av att det följer kodstandarder. Ett JSON meddelande byggs upp av ett JSON-objekt som innehåller JSON-objekt eller JSON-array. Det som finns i JSON-array eller -objektet är olika värden av olika variabeltyper [8].

2.6 Kontinuerligt Lyssnande

Kontinuerligt Lyssnande, mer känt som Continuous Listening, är en idé om att efterlikna hur en människa lyssnar. Mer specifikt lyssnar människor efter vissa utlösare för att starta en konversation, till exempel 'Hej' (människan använder sig utöver ljud även sin syn för att avgöra om en konversation ska initieras). När kontinuerligt lyssnande ska implementeras i kod används mänskligt lyssnande och konverserande som modell. Detta medför att man vill hålla fördröjning mellan förfrågan till respons till ett minimum, då en människa i en konversation ger respons inom loppet av någon sekund.

Ljudsegmentationsalgoritm

En ljudsegmentationsalgoritm är konstruerad för att urskilja ljud från tystnad men kan även användas för att urskilja tal från andra typer ljud i små partier av ljudsignaler. Detta medför att ljudfiler kan göras minimala i storlek. Det finns flertalet välkända algoritmer för att behandla ljudsignaler på sådant sätt.

Zero Crossing Rate (ZCR)

Zero Crossing Rate Zero Crossing Rate (ZCR) är en algoritm som summerar antalet gånger ljudsignalens amplitud passerar nollvärdet under en tidsperiod, sk fönster . Ljudsignalen digitaliseras via en AD-omvandlare med viss samplingsfrekvens. Algoritmen tar in en ljudsignal i form av ett visst antal sampels som den sedan delar upp i flertalet fönster eller ljudpartier. Fönstrets storlek beror på ljudsignalens frekvens och en förbestämd fönsterlängd.

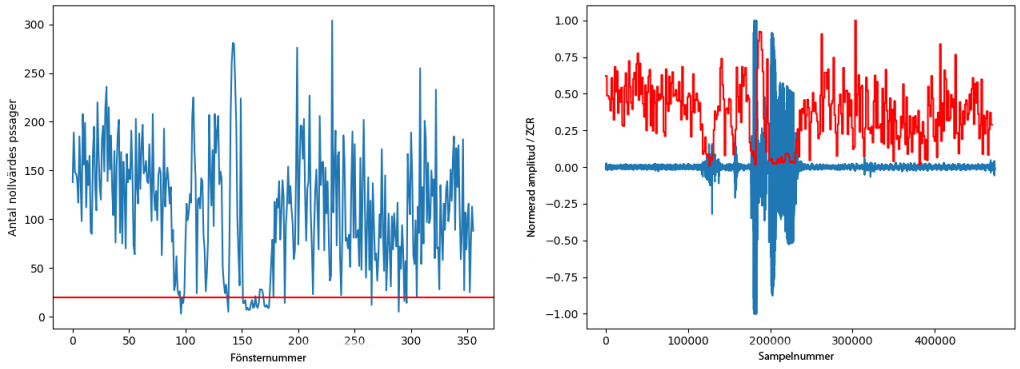
$$fönsterstorlek \approx fönsterlängd \times frekvens$$

Fönstrets storlek beror på ljudsignalens frekvens och en förbestämd fönsterlängd. Antalet fönster som får plats under loppet av signalen beräknas genom att ta antalet sampels i signalen (*signaldata*) dividerat med fönsterstorleken. Värdet trunkeras eftersom algoritmen inte är applicerbar på halva fönster.

ZCR beräknas sedan som antalet nollgenomgångar i ett fönster. ZCR kan anges på olika sätt antingen som ett absolut tal, dvs antal nollgenomgångar i fönstret eller som ett normerat tal där man dividerar antal nollgenomgångar med antal sampels i fönstret. I det senare fallet blir det ett värde mellan 0 till 1 och i det första fallet är värdena mellan 0 och lika med antalet sampels i fönstret.

$$AntalFönster = \lfloor signaldata \div fönsterstorlek \rfloor$$

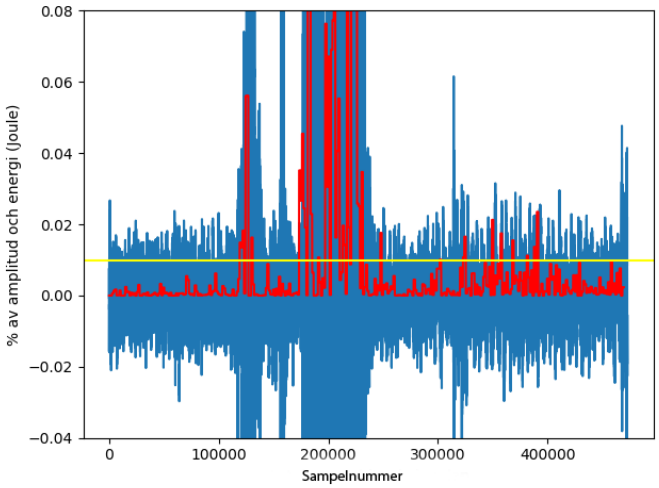
De ljudfönster som innehåller tal har lägre ZCR än fönster utan tal. I figur 2.1 visualiseras ZCR's framgång med att segmentera ljudfiler[9], [10].



Figur 2.1: I den första bilden visas antalet nollgenomgångar, ZCR i varje fönster. Den röda linjen visar tröskeln för ljud som saknar tal (över linjen). Den andra bilden visar värdet på ZCR i varje fönster under loppet av ljudsignalen. ZCR och amplitud mäts här i ett normerat värde mellan 0 och 1.

Short Term Energy (STE)

Algoritmen *Short Term Energy* (STE) bygger på amplituden i en ljudsignal. Generellt sätt så är amplituden för en signal med tal märkbart högre än för signaler utan tal. På dessa variationer i amplitud byggs energidelen av algoritmen upp. Ljudsignalen delas, precis som för ZCR, upp i fönster och energi nivåerna för varje fönster analyseras. På så sätt kan talsignaler skiljas från signaler utan tal. I figur 2.2 visas var gränsen för ljudsegmentationen i STE går [11], [12].



Figur 2.2: Bilden visar energivärdet i varje fönster under loppet av ljudsignalen. Det blåa representerar originalljudet, det röda energi nivåerna och den gula linjen är gränsvärdet. Då originalljudet och energinåvån samtidigt överstiger gränsvärdet sparas ljud i andra fall tas ljudet bort.

3
Metod

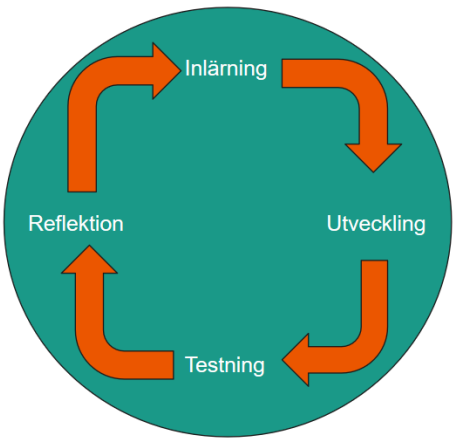
Projektet har utvecklats med influenser från den iterativa utvecklingsmetoden scrum och har visuellt uttryckts med hjälp av ett så kallat scrumboard. För versionshantering av kod har Git använts. Under projektet har dagliga stand-upmöten inträffat, då det är rutinen på Innovation Zone hos Cybercom Group där projektet utförts.

Till en början kretsade arbetet kring inlärning av Python och förståelse för kontinuerligt lyssnande. Med hjälp av handledare på Cybercom togs en initial systemdesign fram. Den har använts som en helhetsbild, med vissa avvikelser, över vilka delar som ingått i projektet.

Då tidigare projekt utnyttjat en NLP-tjänst som, det diskuterats, saknade viss önskad funktionalitet beslutades att en undersökning av NLP-tjänster skulle genomföras. Sammanfattning av tidigare projekt kan finnas i bilaga A.1. När den NLP-tjänsten som lämpade sig bäst för projektet valdes fortskred implementation av kod, inhämtning av information och testning parallellt.

3.1 Den iterativa processen

Den iterativa processen har bestått av fyra huvuddelar som illustreras i figur 3.1 , inlärning, utveckling, testning och reflektion. Man har börjat med att komma fram till vilken uppgift som behöver utföras och på ett scrumboard satt upp det i kategorin över pågående uppgifter. När en uppgift satts som pågående så körs processen igång. Man börjar med att läsa teori om ämnet, eventuellt prata med handledare eller ta onlinekurser, sedan utvecklas och testas mjukvaran agilt. Efter projektgruppen anser att mjukvaran är fungerade så reflekterar gruppen över om den är bra, man tar även upp framsteget med produktägaren.



Figur 3.1: Den iterativa process som använts.

3.2 Minsta Möjliga Produkt (MVP)

En MVP togs fram genom en, till en början, intern diskussion om vad som var av intresse att utveckla. I senare stadie hölls även diskussion med beställare om vad som var viktigt för dom. I slutändan innehåller MVP att ChatUp vid uppstart startas i ett läge där den kontinuerligt lyssnar efter frågor eller kommandon och därefter ha förmågan att:

- Svara på frågor inom förbestämda kategorier.
- Småprata om ämnen som inte faller inom de förbestämda kategorier.
- Svara användaren med hjälp av röst.
- Ha en responstid på under 5 sekunder då förfrågan är under 5 sekunder.

3.3 Ytterligare Funktionalitet

Chatup utvecklades först till en MVP-version och ytterligare funktionalitet utvecklades sedan iterativt. Funktioner valdes med hänsyn utifrån lämplighet, användarpotential och intresse. Följande funktionalitet är ej listat i ordning.

- Startas i ett standbyläge där ChatUp kontinuerligt lyssnar efter hälsningsfraser.
- Efter avslutad konversation gå tillbaka till att kontinuerligt lyssna efter hälsningsfraser.
- Mer komplex chatbotfunktionalitet.

3.4 Testning

Då resultatet är en chatbot som man kan prata med och målet har varit att sänka responstiden så mycket som möjligt har testning skett kontinuerligt enligt den agila arbetsmetod vi arbetat efter. Testning utfördes genom att ställa frågor direkt till ChatUp och mäta responstiden. Resultatet av mätningarna kan sedan användas för att besluta om en viss algoritm är bättre än en annan. För att visualisera de mätningar som används i rapporten har pyplot, ett Python-bibliotek som används för att rita grafer, använts.

4
Roboten Nao med befintlig mjukvara

I detta avsnitt kommer vi gå igenom Nao-roboten vi jobbat med samt den mjukvaran som redan var installerad sedan tidigare projekt.

4.1 Nao-robotens hårdvara

Ursprunglig hårdvara i Nao-roboten:

- Förmågan att röra på sig, med tillräcklig finmotorik för att dansa Macarenan
- Förmågan att spela upp ljud
- En ethernetport som används för att skicka kommandon in till enheten och kontrollera den.

Ytterligare hårdvara som lades till i efterhand innan detta projekt:

- En Raspberry Pi som placerats i en specialdesignad 'ryggsäck' gjord för att bära den där all mjukvara är installerad.
- En powerbank som fästs med kardborreband som är strömkällan till Raspberry Pi:n
- En mikrofon för att kunna spela in ljud.

4.2 Mjukvara installerad på roboten

Det tidigare projektet som har:

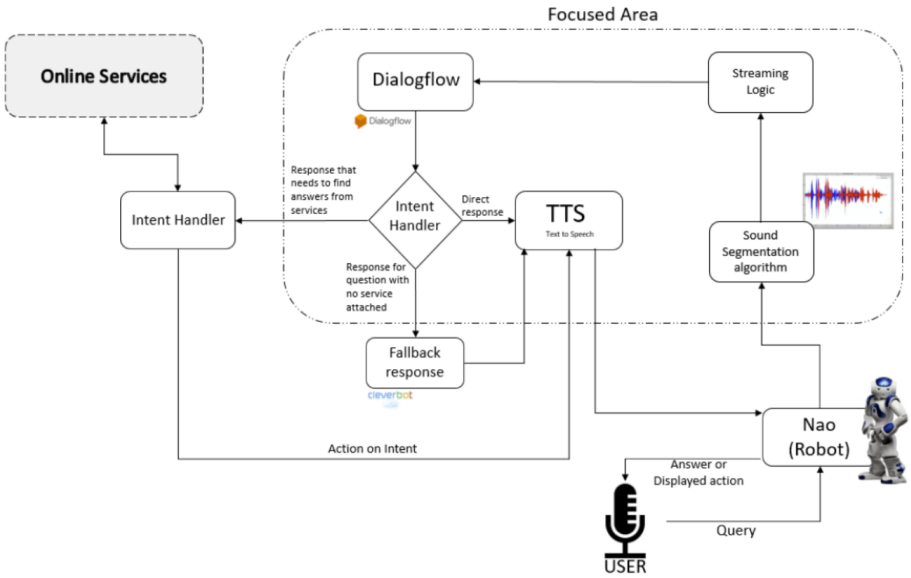
- Filer med rörelsekommandon till roboten och logik för att skicka dessa som kommandon till Nao-enheten
  - En sekvens rörelser vilket resulterar i att roboten vinkar.
  - En sekvens rörelser vilket resulterar i att roboten dansar Macarenan.
- Logik för att strömma data till en molnbaserad server.
- Logik för att känna igen ansikten med hjälp av kameran.

5
Genomförande

Initialt togs en systemdesign fram, Figur 5.1, där man kan följa flödet av information i systemet. I följande avsnitt kommer varje del av designen beskrivas, där fokusområdet kommer beskrivas i detalj.

5.1 Systemdesign

Grundprincipen hos ChatUp är att en användare via röst ger ett kommando eller ställer en fråga. Förfrågan strömmas sedan i realtid till en NLP-tjänst som tar fram dess avsikt. När avsikten hittats hanteras den och skickas vidare till rätt service. I några fall har en avsikt ett direkt svar knutet till sig och i dessa fall skickas svaret direkt till en *text to speech* (TTS) som via roboten spelar upp svaret för användaren. Finns det inget knutet svar till avsikten så skickas frågan till en *fallback* som via TTS spelar upp svaret för användaren. På samma sätt som fallback fungerar även andra tjänster för svar till exempel Västtrafik för svar om busstider och Google för svar om restauranger.



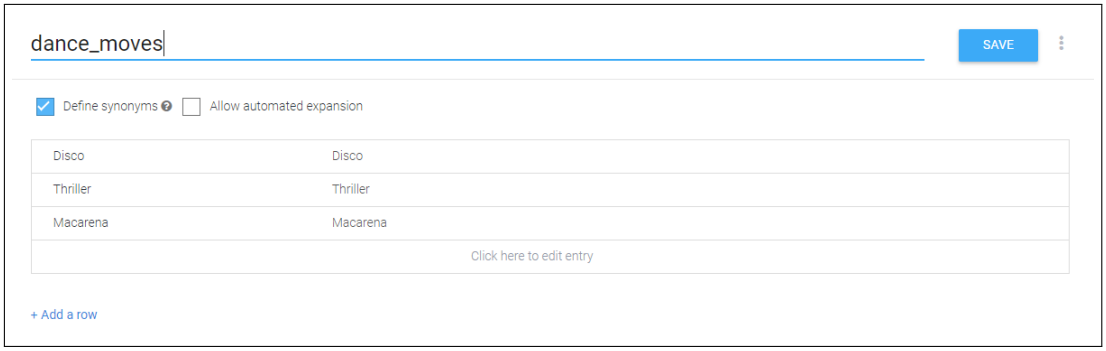
Figur 5.1: Karta över systemdesignen med avgränsat fokus område.

Designen togs fram genom att diskutera med produktägarna om vad som var kritiskt och sedan diskutera inom projektet om hur man bäst skulle realisera deras vision. Det finns flertalet alternativa fokusområden som hade varit lika förmånliga för produktägarna till exempel utveckla hur Nao-roboten fungerar, då till och med utvecklaren som skrivit den koden anser att den kunde gjorts mycket bättre. Detta valdes dock inte då beställare tänkt köpa in en ny robot kallad Pepper istället för Nao.

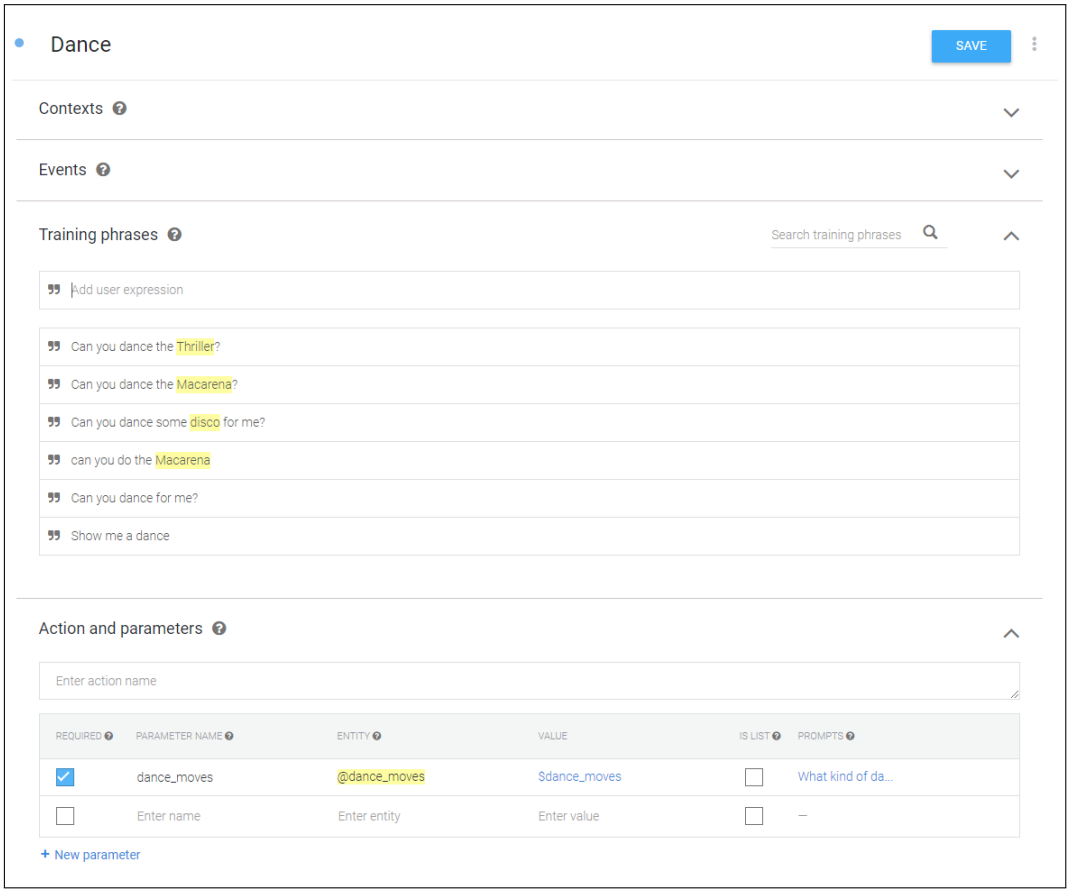
5.2 Skapandet av chatboten

Projektet startade med att undersöka NLP-tjänster och se vilken som var lämpligast för detta projektet. Produktägaren hade tidigare dåliga erfarenheter av Amazon's Lex och vill därför att vi skulle undersöka andra alternativ. I ett tidigare projektarbete så användes Wit.ai, en NLP-tjänst vars styrkor ligger i åtkomligheten och antalet språk som stöds. Dock så har den flera svagheter som gjorde att man inte valde att fortsätta med den i detta projektet. De svagheter som låg till störst grund till valet var att Wit.ai inte har varken stöd för kontextbaserade svar eller förmågan att strömma ljud kontinuerligt direkt till tjänsten. Dialogflow undersöktes och till skillnad från Wit.ai så är den mer komplex och svårare att använda men är därefter ett mer kraftfullt verktyg som kan ställa frågor tillbaka till användaren och använda deras respons för att ta reda på användarens avsikt. Dialogflow har nackdelen att den (då detta projektet startades) inte stödjer svenska men den nackdelen sågs inte överväga fördelarna. Därför gjordes kompromissen att kommandon och förfrågningar endast skulle utvecklas och tas emot på engelska.

I Dialogflow så utvecklades chatboten genom att träna den med flera olika fraser och meningar angående varje kategori, t.ex. för att få den att kunna förstå att man vill att den ska dansa så skapades först en entitet kallad 'dance_moves' som man ser i figur 5.2. Denna innehöll alla typer av danser roboten skulle kunna utföra. Efter det så skapade vi en avsikt som kallades 'Dance' där vi matade in flera olika fraser om hur en användare kan fråga efter en dans vilket man ser i figur 5.3. Detta gjordes iterativt med allt mer och mer funktionalitet. I vissa fall som med 'Dance' så gjorde vi entiteten 'dance_moves' obligatorisk, det vill säga att om en användare endast frågar om en dans utan att specificera vilken typ av dans så kommer Dialogflow be användaren att svara med en typ av dans.



Figur 5.2: Entiteten 'dance_moves'



Figur 5.3: 'Dance'-avsikten med några träningsfraser

5.3 Strömmningslogik

För att kunna strömma ljud till programmet så har PyAudio använts då det är väldigt simpelt att använda och väldigt lämpat för live-strömmningar. För att kunna avgöra när användaren har slutat prata med enheten så har WebRTC's VAD (*Voice Activity Detector*) [15] använts för att analysera ljudet. VAD:n tar fram ifall ett stycke innehåller tal eller ej, om det finns tal i stycket skickas det vidare till Dialogflow som vidare analyserar ljudet för att plocka ut avsikten och eventuella entiteter. Vi valde denna VAD eftersom den har vart till hjälp i ett tidigare projekt på företaget och rekommenderades av vår handledare. Skillnaden mellan WebRTC's VAD och våra ljudsegmentationsalgoritmer är att VAD:n endast analyserar ljud och returnerar ett booleskt värde baserat på resultatet medan algoritmerna manipulerar signalen och skickar tillbaka delarna som upplevs innehålla tal. Om VAD:n inte upptäcker tal avbryts strömningen till Dialogflow för att effektivisera processen och minimera responstiden. För att simulera effekten att roboten endast svarar när man pratar med den så har två lägen skapats som den konstant befinner sig i, stand-by och aktivt lyssnande. När den är i stand-by så ignorerar den allt man säger om inte det är en hälsningsfras av något slag. När den är i aktivt lyssnande så svarar den på allt som den uppfattar tills man säger en avskedsfras och då går den tillbaka till stand-by-läge tills man startar upp den igen med en hälsningsfras.

5.4 Handling baserat på avsikt

Som man ser i bilaga A.2 så får man väldigt utförlig information om vad användaren har sagt och deras avsikter, i JSON-meddelandet så kan man plocka ur nyckelord som används för att välja vilken tjänst och respons vi vill ge användaren. Ett exempel på en sådan tjänst är Cleverbot. Ett utav målet som sattes för projektet var att roboten ska kunna småprata med användaren och alltid ha ett svar av viss relevans, med detta i åtanke så är Cleverbot en perfekt tjänst. Detta då den har specialiserat sig på exakt småprat och det skulle vara alldeles för tidskrävande att ta fram en egen lösning då den inte hade kunnat bli lika bra under den relativt korta tidsperioden. I ChatUp används Cleverbot som en reservutväg när det är oklart vad användaren vill. När det bestämts att Cleverbot ska användas skickas en förfrågan till Cleverbots API som i sin tur skickar ett svar tillbaka till ChatUp. Andra tjänster som utnyttjas är L's Kitchens API för att ta reda på dagens lunch, Chuck Norris-skämtdatabasen och en ordvitsdatabas för att kunna berätta skämt då de hade ett enormt utbud och variation. För att få roboten att dansa så utnyttjar vi redan befintliga funktioner som skickar kommandon till roboten med enskilda rörelser som blir en dans när de exekveras på följd. Den valda dansen är Macarenan då den är lätt igenkännlig och framkallar nostalgi vilket ytterligare ökar intresset med roboten.

När ett svar mottagits skickas det vidare som en textsträng till en TTS. ChatUp använder gTTS som är en TTS från Google detta då den är lätt och implementera samt att den ger önskvärda resultat.

5.5 Ljudsegmentationsalgoritm

Utvecklingen av ljudsegmentationsalgoritmerna utgick från att leta upp kända algoritmer som undersökts och använts för andra typer av projekt. Ett beslut fattades att undersöka två tillvägagångssätt mer noggrant det ena bygger på volymen i ljudsignalen medan den andra bygger på Markov-kedjor. Då Markov-kedjor bygger på statistik där man räknar på sannolikheten av att något händer och sedan försöker förutspå vad som ska hända nästa gång behövs många datapunkter. Eftersom undersökningen av algoritmerna började sent och projektet har en begränsad tid samt att kunskap inom markov-statestik var bristande beslutades det att markov-kedjor inte skulle användas.

Då Markov-kedjor föll bort undersöktes istället två nya algoritmer *Zero Crossing Rate (ZCR)* och *Short Term Energy (STE)*. Dessa algoritmer verkade initialt som snabba algoritmer som kan utföra segmentationen av ljud med bra resultat. Då många tidigare skrivit dessa algoritmer i MATLAB användes den kod som hittats som grund för algoritmerna i ChatUp. Koden som hittades har optimerats och förändrats för att passa ChatUps specifika syfte. Detta då man traditionellt inte använt dessa algoritmer för att segmentera ljud i realtid utan endast färdiga ljudsignaler [13], [14].

5.6 Testning

ChatUp har testats med många frågor från flertalet personer för att testa olika funktioner. Både Fallback och Online Services, se Figur 5.1, har testats. Mjukvaran har även installerats på Nao-roboten och testats där. Algoritmernas effektivitet har testats med en ljudinspelning som yttrar frasen;

"What's the capital of France?"

Algoritmernas förmåga att behålla kvalitén på ingående förfrågningar har även testats i live-strömmningsläge där en person yttrat ett antal olika fraser av varierande längd och komplexitet vilka listas i tabell 5.1. Ett stickprov genomfördes då testning tidigare har genomförts omgående utan dokumentation med liknande resultat.

Tabell 5.1: Fraser som testats i live-strömmningsläge

Yttrad fras
hey
how are you doing
are you planning a vacation soon
tell me a joke
show me a dance

Med hjälp av ljudinspelningarna och live-strömningen har responstiden kunnat mätas och efter analys kan man då ta fram den mest effektiva algoritmen för ChatUp.

Mätningarna har genomförts genom att en timer startats när en förfrågan yttrats och sedan stoppats när ChatUp givit sin respons.

Tester genomfördes även på live-strömmningsläget i syfte att mäta responstiden utan manipulerande algoritmer genom att testa två fraser under 5 sekunder och mäta ChatUps responstid.

6
Resultat

Projektets mål anses uppnått, dock har ChatUps kunskap värderats som mindre viktigt då konceptet av hur man går tillväga för att förstora ChatUps kunskapsbas räcker.

En fungerande chatbot har skapats och integrerats i den fysiska roboten. För att använda programmet i roboten så initierar man först programmet och sen så ger man en hälsningsfras till robotten för att få den att börja lyssna på ens förfrågningar. Därefter kan man ge den förfrågningar tills man ger den en avskedsfras då den går tillbaka till stand-by.

6.1 Uppnådd funktionalitet

ChatUp har uppnått de krav som beskrivs av målen, under rubriken 1.3 Mål, och MVP:n, under rubriken 3.2 Minimum Viable Produkt, och har utöver det utvecklats med ytterligare funktionalitet.

Mål som uppnåddes

- ChatUp lyssnar kontinuerligt på alla förfrågningar.
- ChatUp svarar i aktivt läge på alla förfrågningar.
- ChatUp ignorerar alla förfrågningar som inte är en hälsnignsfras när den är i stand-by-läge för att simulera effekten av att den endast svarar på tilltal.
- ChatUps responstid har kortats ner avsevärt och är under 5 sekunder då förfrågan också är det.

Mål som delvis uppnåddes

- ChatUp har tre kategorier och en fallback-kategori.

Ytterligare funktionalitet som implementerats

- ChatUp lyssnar kontinuerligt på alla förfrågningar efter hälsningsfraser.
- ChatUp går efter avslutad konversation tillbaka till att kontinuerligt lyssna efter hälsningsfraser.
- Mer komplex chattbotfunktionalitet lades till i form utav kontextbaserade svar och förmågan att chattboten minns den tidigare förfrågan.

6.2 Resultat av algoritmernas effektivitet

Nedan följer de resultaten från responstidsmätningarna som är relevanta för vår slutsats. Resterande resultat från mätningarna kan finnas i appendix A.2.

6.2.1 Ljudfilsströmmning

Med test mot en ljudfil kan man observera att ZCR-algoritmen är den algoritm som påverkar svarstiden märkbart, figur 6.1. När STE-algoritmen används kan man däremot endast se marginell förändring från att inte utnyttja någon algoritm, figur 6.2 jämtemot figur 6.3.

```
1.8136720657348633
1.7885019779205322
1.6730027198791504
1.604048252105713
1.869687795639038
1.6171824932098389
1.6491599082946777
1.6751024723052979
1.7714695930480957
1.6466538906097412

mean: 1.7108481168747
```

Figur 6.1: Mätning med ZCR. Alla resultat är i sekunder..

```
1.8144927024841309
2.3240013122558594
2.091078281402588
2.0781359672546387
2.1284103393554688
2.1106467247009277
1.917569875717163
2.0987892150878906
2.10334849357605
2.1629538536071777

mean: 2.0829426765442
```

Figur 6.2: Mätning med STE. Alla resultat är i sekunder.

```
2.012955665588379
2.1493844985961914
2.0581321716308594
2.1589550971984863
2.1772143840789795
2.0120205879211426
2.09946870803833
2.141035318374634
1.8987011909484863
2.031919002532959

mean: 2.0739786624908
```

Figur 6.3: Mätning utan manipulation. Alla resultat är i sekunder.

6.2.2 Live-strömmning

I live-strömmningsläge strömmas ljudet direkt från en mikrofon. Algoritmerna har i detta läge en avsevärd svaghet då de inte alltid behåller kvaliteten hos användarens förfrågningar och mer än ofta tar bort viktiga delar av meningen, vilket leder till att Dialogflow inte förstår vad användaren sagt.

ZCR-algoritmen förstår kortare och mindre komplexa meningar bra men misslyckas när komplexiteten och längden går upp, se tabell 6.1. STE-algoritmen däremot är väldigt dålig på att behålla meningar i dess rätta format när man strömmer ljudet direkt och klarar endast enskilda ord utan att misslyckas med att förmedla vad det var användaren ville säga, se tabell 6.2. När algoritmerna används blir inspelningstiden väldigt statisk vilket gör att långa fraser kan bli avklippta eller att korta fraser får en längre responstid än nödvändigt. Utan manipulation kan däremot kvalitén behållas, se tabell 6.3.

Tabell 6.1: ZCR-algoritmens försök att behålla kvalitén på ingående fras

Yttrad fras		ChatUps tolkning
hey	->	hey
how are you doing	->	how are you doing
are you planning a vacation soon	->	polygon navigate
tell me a joke	->	so let me know
show me a dance	->	show women

Tabell 6.2: STE-algoritmens försök att behålla kvalitén på ingående fras

Yttrad fras		ChatUps tolkning
hey	->	hey
how are you doing	->	how are you
are you planning a vacation soon	->	i'm going to leave
tell me a joke	->	holyoke
show me a dance	->	Madonna

Tabell 6.3: Förmågan att behålla kvalitén på ingående fras utan manipulation

Yttrad fras		ChatUps tolkning
hey	->	hey
how are you doing	->	how are you doing
are you planning a vacation soon	->	are you planning a vacation soon
tell me a joke	->	tell me a joke
show me a dance	->	show me a dance

När test emot live-strömmning utan manipulerande algoritmer avseende responstid genomfördes kan en lägre responstid observeras i jämförelse med det tidigare projektet på Cybercom då man hade en statisk inspelningstid på fem sekunder och målet som sattes upp i 1.3.

Tabell 6.4: Responstiden av live-strömning utav ljud med två olika fraser och VAD. Alla resultat är i sekunder.

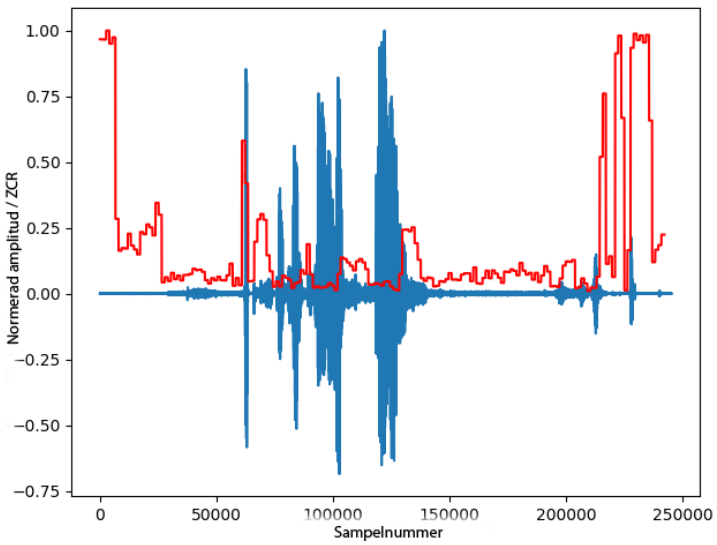
	"Hello"	"How are you doing?"
	2.9300143718719482	4.169834613800049
	3.511841297149658	4.328438758850098
	3.5573785305023193	4.183203458786011
	4.807077884674072	4.06703519821167
	4.063820123672485	3.9206459522247314
Mean value:	3.7740264415741	4.1338315963745

7
Analys

Utvecklingen av en social robot anses vara uppfyllt i form av en mjukvarumodul; en plattform för att vidareutveckla och experimentera inom maskinintelligens. Med hjälp av ljudsegmentations algoritmer och talanalys har responstiden från förfrågan till svar minskat avsevärt. ChatUp anses därför ha nått en mer sömlös interaktionsprocess än tidigare.

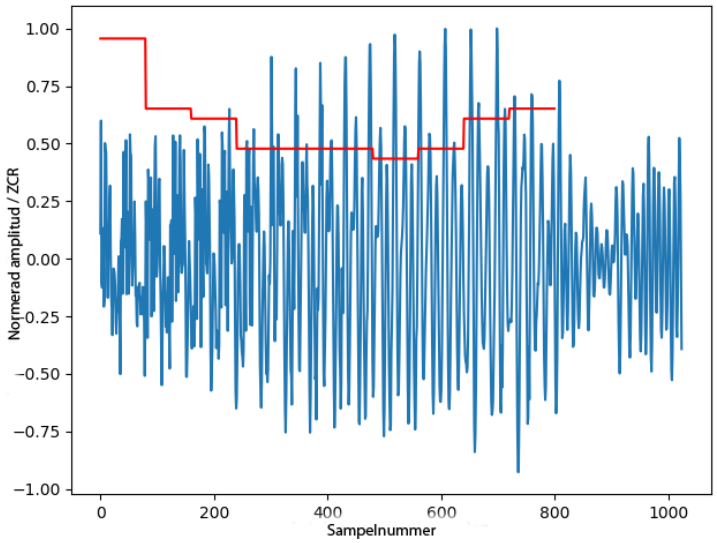
7.1 Angående algoritmernas effektivitet

Som man kan se i resultatet så funkar ZCR-algoritmen väldigt bra över en hel ljudfil men mycket sämre när det gäller live-strömmning av ljud. Detta beror på att när man applicerar algoritmen på en hel fil så kan den verka på större fönsterstorlek vilket leder till en mycket jämnare applicering vilket man ser i figur 7.1. Man kan se att den normerade ZCR är mycket låg kring segment med hög amplitud vilket innebär att talet blir inkluderat.



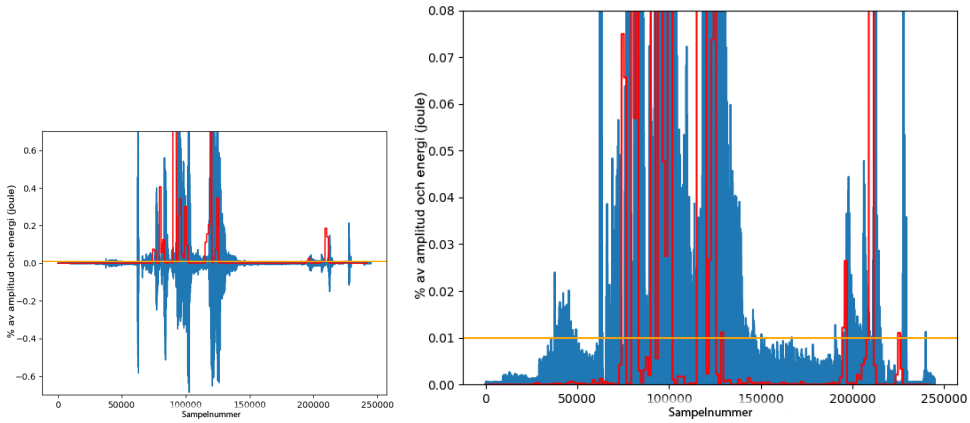
Figur 7.1: ZCR applicerat över en ljudfil där den röda linjen är normerad ZCR och det blå är normerad amplitud på signalen. Låga värden på ZCR innebär att ljudet betraktas som ljud och inkluderas.

När algoritmen får in kortare ljudsignaler som vid gradvis strömmande av ljud påverkar det noggrannheten och ZCR tar även bort fönster av ljud som innehåller tal. I figur 7.2 visar att inget av ljudet i filen betraktas som ljud då ZCR är väldigt högt för hela filen, detta gör att det totala ljudet som skickas till Dialogflow är väldigt fragmenterat och vissa delar saknas. Avsaknandet av vissa ljudbitar resulterar i att den inte förstår avsikten som användaren har.

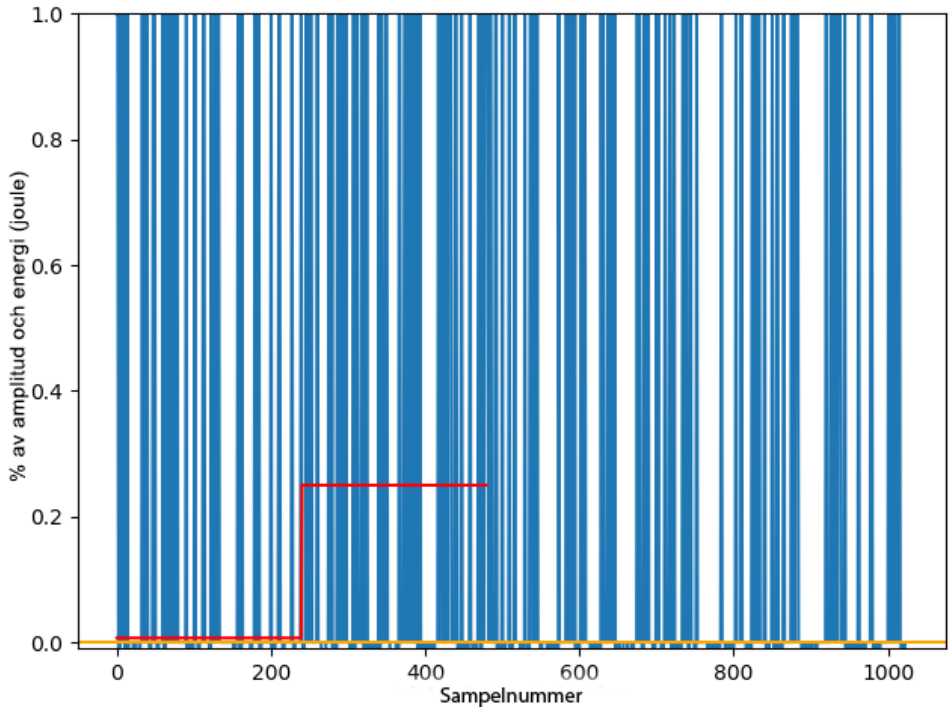


Figur 7.2: ZCR applicerat över en väldigt liten del av en ljudström där allt blått är normerad amplitud av signalen och den röda linjen är normerad ZCR där höga värden innebär att ljudet betraktas som icke-tal och inkluderas ej.

I STE-algoritmen ser man en väldigt liknande effekt där den är otroligt effektiv och noggrann på ett stort spann, så som en ljudfil vilket man ser i figur 7.3. Medan den över ett litet stycke av ljud inte klarar av att mäta energin över så få bitar per fönster, vilket man ser i figur 7.4 där den endast lyckas inkludera knappt hälften av ljudet.



Figur 7.3: STE applicerat över en ljudfil där allt blått är normerat amplitud av signalen, det röda är den uppmätta normerade energinivån och den gula linjen är tröskeln för vad som uppfattas som tal. Allt ljud markerat av den röda linjen som är över tröskeln klassas som tal.



Figur 7.4: STE applicerat över en väldigt liten del av en ljudström där allt blått är normerat amplitud av signalen, det röda är den uppmätta normerade energinivån och den gula linjen är tröskeln för vad som uppfattas som tal. Allt ljud markerat av den röda linjen som är över tröskeln klassas som tal.

7.2 Angående responstid

Trots algoritmernas motgångar under live-strömmning av ljud har responstiden ändå lyckats kortas ner, vilket medfört att roboten har blivit mycket mer responsiv. Detta genom att analysera ljudet istället för att manipulera det. När VAD används för att undersöka varje parti av ljudströmmen och kommer fram till om den innehåller tal eller inte. Detta tillskillnad från algoritmerna som försöker ta bort partier av ljud som inte innehåller tal.

När programmet upplever att användaren slutat tala avbryts ljudströmmen. Detta har minskat responstiden markant då det tidigare användes inspelningsperioder på fem sekunder. Inspelningsperioden ledde till att responstiden aldrig var under fem sekunder samt att längre fraser avbröts och att användaren behövde vara väldigt precis i sin tajmning när roboten lyssnade. Med denna lösningen kan roboten lyssna konstant och avbryter endast när en användare slutat prata med den.

Detta har gjort roboten mycket mer responsiv och har en mycket större förmåga att ha konversationer som upplevs mänskliga.

7.3 Angående att endast lyssna på tilltal

ChatUp klarar av att lyssna enbart på den som tilltalar den endast under begränsade förhållanden. Efter hälsningsfras yttrats lyssnar den på allt tills man yttrar en avskedsfras, som stänger av lyssnandet. Det görs ett antagande att folk inte hälsar på varandra i närheten av roboten. Det är möjligt att folk pratar runt den och att den plockar upp tal som den inte borde men så länge användare kommer ihåg att ge den en avskedsfras kommer irrelevant tal att begränsas. ChatUp har dock inte förmågan att göra separationer om folk pratar med den eller inte utan är endast begränsad till förhållanden där sannolikheten för tal som är riktat till roboten är stor.

7.4 Angående kunskap inom olika kategorier

Målet att ChatUp skulle behärska kunskap inom sex olika kategorier nåddes inte då målet modifierats under processens gång. Efter återkoppling från produktägarna prioriterades detta målet ner till en konceptvalidering som utvecklats genom att återanvända två kategorier som använts tidigare i roboten. Utöver de två kategorierna har en kategori som behandlar frågor om matutbud implementerats. De tre kategorier som utelämnats var:

- Lokalisera var i byggnaden ett specifikt rum ligger.
- Kolla upp om en viss person befinner sig på möte eller på sitt kontor.
- Ta fram nästa buss, spårvagn eller färja som går.

8
Diskussion

8.1 Resultat och analys

Det är intressant hur de olika algoritmerna skiljer sig åt i effektivitet under liveströmmning jämnfört med när de körts på en ljudfil. Upptäckten av algorimternas bristande förmåga att behålla kvalité och valet att byta sätt att tackla problemet på har gjort ChatUp till ett mer användarvänligt system.

För att ChatUp i framtiden ska vara användbar behövs det mycket vidareutveckling med fler kategorier. Då målet är att ChatUp ska uppfattas så mänsklig som möjligt kan framförallt analysen av ljud förbättras men även hårdvara som förmedlar att den är någon som man kan prata med.

8.2 Miljö och Etik

ChatUp är en chatbot med en stark basfunktionalitet och har stark utvecklingspotential. Detta gör att additionen av fler online-tjänster för att ta reda på diverse olika specifika förfrågningar kan realiseras snabbt. Detta innebär i sin tur att utvecklare kan komplettera ChatUp med fler möjligheter för människor att välja ett mer klimatsmart liv. ChatUp har redan nu en tjänst som kollar upp matutbudet i området. Eftersom det då är lättare att ta reda på vad det finns för mat i närheten gör detta att valmöjligheten blir större och det kan innebära att fler miljömedvetna matval görs.

En risk med att utveckla allt mer sociala robotar är att mänsklig interaktion eventuellt kan bytas ut mot robotar om upplevelsen simulerar mänskliga konversationer tillräckligt bra. Något annat att ha i åtanke är att om robotar kan simulera mänskliga interaktioner och känslor tillräckligt bra, var går gränsen för vad som kan klassas som medvetande? Innebär detta då att vi människor i framtiden kommer behöva ha någon form av etiskt ansvar för robotorna?

8.3 Övriga funderingar

Under projektets gång har en undersökning av flera olika segmentationsalgoritmer och algoritmer för taligenkänningsanalys gjorts. Den algoritm som verkade vara bäst för det som ChatUp ska göra är byggd på markov modeller. På grund av bristande kunskap inom matematiken som utgör markov modeller och att projektet varit tidsbegränsat blev implementationen av dessa en omöjlighet.

Det hade varit lättare att genomföra projektet om det funnits en klar bild från produktägarna vad som var prioritet i deras ögon. Om projektets scope varit klart från början fanns det mycket bättre möjligheter att implementera till exempel markov modeller och på så sätt effektivisera taligenkänningsanalysen.

9
Förslag till fortsatt arbete

Eftersom hela projektet har haft som grundtanke att skapa något för vidareutveckling har det inte varit svårt att tänka sig saker man skulle kunna fortsätta med.

- Stöd för robotens seende, man kan tänka sig att roboten kan ta reda på om någon kollar på den och i sådana fall initiera en konversation istället för att vänta på att någon ska initiera konversationer med den.
- Effektivisera taligennkänningsanalysen med hjälp av tillexempel Markov modeller, med markov modeller kan man urskilja olika talare.
- Stöd för att bedöma användarens position i förhållande till roboten.
- Utöka chatbotens funktionalitet genom att addera fler onlietjänster.
- Implementera en egen speciellt anpassad maskininlärningsmodul.
- Implementera fler språk för ChatUp att hantera.
- Undersöka Google duplex integration.

Litteraturförteckning

[1] C. Johnson, 'Programmering är framtidens språk' *Svenska Dagbladet*, Oktober 1, 2013. [Online]. Tillgänglig: www.svd.se/programmering-ar-framtidens-sprak [Hämtad: Juni 1, 2018]

[2] 'Karriärsajt ska locka fler unga till IT-yrken' *Mynewsdesk*, Mars 17, 2011. [Online]. Tillgänglig: www.mynewsdesk.com/se/pressreleases/karriaersajt-ska-locka-fler-unga-till-it-yrken-599312 [Hämtad: Juni 1, 2018]

[3] 'What is Python? Executive Summary' *Python*, n.d. [Online]. Tillgänglig: www.python.org/doc/essays/blurb/ [Hämtad: Maj 22, 2018]

[4] H. Pham, 'Pyaudio Documentation' *Massachusetts Institute of Technology*, n.d. [Online]. Tillgänglig: people.csail.mit.edu/hubert/pyaudio/docs/ [Hämtad: Juni 1, 2018]

[5] 'What is Natural Language Processing?' *SAS*, n.d. [Online]. Tillgänglig: https://www.sas.com/en$_u s/insights/analytics/what-is-natural-language-processing-nlp.html$ [Hämtad: Juni 12, 2018]

[6] 'Dialogflow Documentation' *Dialogflow* , n.d. [Online]. Tillgänglig:dialogflow.com/docs/ [Hämtad: Juni 12, 2018]

[7] 'About Cleverbot' *Cleverbot*, n.d. [Online]. Tillgänglig:www.cleverbot.com [Hämtad: Juni 12, 2018]

[8] 'Introducing JSON' *JSON*, n.d. [Online]. Tillgänglig: https://www.json.org/ [Hämtad: Juni 1, 2018]

[9] M. Jalil, F. Awais Butt och A. Malik Short-Time Energy, Magnitude, Zero Crossing Rate and Autocorrelation Measurement for Discriminating Voiced and Unvoiced segments of Speech Signals"*IEEE Xplore*, 2013 [Kapitel #2, paragraf C] ieeexplore.ieee.org/document/6557272/ [Hämtad: April 23, 2018]

[10] Bachu R.G., Kopparthi S., Adapa B. och Barkana B.D., 'Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal' *Electrical Engineering Department at School of Engineering, University of Bridgeport* [Online]. Tillgänglig: www.asee.org/documents/zones/zone1/2008/student/ASEE12008$_0$044$_p$*aper.pdf*

[11] M. Jalil, F. Awais Butt och A. Malik Short-Time Energy, Magnitude, Zero Crossing Rate and Autocorrelation Measurement for Discriminating Voiced and Unvoiced segments of Speech Signals"*IEEE Xplore*, 2013 [Kapitel #2, paragraf A] ieeexplore.ieee.org/document/6557272/ [Hämtad: April 23, 2018]

[12] Bachu R.G., Kopparthi S., Adapa B. och Barkana B.D., 'Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal' *Electrical Engineering Department at School of Engineering, University of Bridgeport* [Online]. Tillgänglig: www.asee.org/documents/zones/zone1/2008/student/ASEE12008$_0$044$_p$*aper.pdf*

[13] http://www.jcbrolabs.org, L006 : Short Term Zero Crossing Rate (ZCR) Silence Removal", *Speech and Audio Signal Processing Using MATLAB*, Juni 5, 2017. [Online] Tillgänglig: www.youtube.com/watch?v=q9nki9ksHHst=1204s [Hämtad: Juni 20, 2018].

[14] http://www.jcbrolabs.org, L007: Short Term Energy (STE) calculation and Silence Removal", *Speech and Audio Signal Processing Using MATLAB*, Juni 12, 2017. [Online] Tillgänglig: www.youtube.com/watch?v=q9nki9ksHHst=1204s[Hämtad: Juni 20, 2018].

[15] 'Python interface to the WebRTC Voice Activity Detector' *wiseman*, Januari 13, 2018. [Online]. Tillgänglig: ithub.com/wiseman/py-webrtcvad [Hämtad: Juni 25, 2018]

A
Appendix 1

A.1 Sammanfattning från tidigare projekt

Varje år så kommer det nya studenter till området Lindholmen. Det finns därmed ett intresse att underlätta för deras studiegång, detta är ChatBots uppgift. Om en student är osäker på vad som finns tillgängligt i området så kan ChatBot till exempel lista restauranger genom en enkel fråga. ChatBot är programmerat att ta emot en fråga, och ge respons på den frågan. Den gör detta genom att använda sig av en NLP som tolkar frågan, varav programmet sedan anropar en specifik service för att kunna hämta information som är relevant. Chatbot är utvecklad i Java med Wit.ai som NLP och är endast textbaserad.

A.2 Json svar från Dialogflow


```
{
  "responseId": "6231fadc-7c9a-4c73-b7d8-e2fa6c515d0e",
  "queryResult": {
    "queryText": "show me a dance",
    "parameters": {
      "dance_moves": ""
    },
    "fulfillmentText": "What kind of dance do you want me
    to do? I can do the Macarena, disco and the Thriller",
    "fulfillmentMessages": [
      {
        "text": {
          "text": [
            "What kind of dance do you want me to do? I can
            do the Macarena, disco and the Thriller"
          ]
        }
      }
    ],
    "outputContexts": [
```

```
      {
        "name": "projects/test-76a5d/agent/sessions/40ce35ec
        -30df-4522-aa82-068a3e70ba46/contexts
        /dance_dialog_context",
        "lifespanCount": 2,
        "parameters": {
          "dance_moves": "",
          "dance_moves.original": ""
        }
      },
      {
        "name": "projects/test-76a5d/agent/sessions/40ce35ec
        -30df-4522-aa82-068a3e70ba46/contexts
        /dance_dialog_params_dance_moves",
        "lifespanCount": 1,
        "parameters": {
          "dance_moves": "",
          "dance_moves.original": ""
        }
      },
      {
        "name": "projects/test-76a5d/agent/sessions/40ce35ec
        -30df-4522-aa82-068a3e70ba46/contexts/
        33d60b6b-2b5a-4d4e-bd6c-4fb97487b589_
        id_dialog_context",
        "lifespanCount": 2,
        "parameters": {
          "dance_moves": "",
          "dance_moves.original": ""
        }
      }
    ],
    "intent": {
      "name": "projects/test-76a5d/agent/intents/33d60b6b
      -2b5a-4d4e-bd6c-4fb97487b589",
      "displayName": "Dance"
    },
    "intentDetectionConfidence": 1,
    "diagnosticInfo": {},
    "languageCode": "en"
  }
}
```


Figur A.1: Json svar från dialogflow

A.3 Mätningar av responstid vid ljudfilsströmmning

Följande mätningar är utförda utan VAD

```
2.054241418838501
2.233884811401367
1.9360895156860352
2.0290536880493164
2.2711386680603027
2.1117279529571533
1.9755816459655762
2.233299732208252
2.4094104766845703
2.0160062313079834

mean: 2.1270434141159
```

Figur A.2: Mätning utan algoritm

```
2.0669381618499756
2.170626401901245
2.0965564250946045
2.5097219944000244
2.2287240028381348
2.2420661449432373
1.999098300933838
2.1423425674438477
2.3886466026306152
2.5805864334106445

mean: 2.2425307035446
```

Figur A.3: Mätning med STE

```
1.7014930248260498
1.7574889659881592
1.6206541061401367
1.553781270980835
1.7209157943725586
1.6849384307861328
1.7177138328552246
1.7643613815307617
1.5436935424804688
1.5895156860351562

mean: 1.6654556035995
```

Figur A.4: Mätning med ZCR

```
1.8249454498291016
1.9753177165985107
1.6231982707977295
1.6057453155517578
1.7125532627105713
1.938633918762207
1.7282476425170898
1.477379322052002
1.7208538055419922
1.7822344303131104

mean: 1.7389109134674
```

Figur A.5: Mätning med ZCR och STE

Följande mätningar är utförda med VAD

```
2.012955665588379
2.1493844985961914
2.0581321716308594
2.1589550971984863
2.1772143840789795
2.0120205879211426
2.09946870803833
2.141035318374634
1.8987011909484863
2.031919002532959

mean: 2.0739786624908
```

Figur A.6: Mätning med bara VAD

```
1.8144927024841309
2.3240013122558594
2.091078281402588
2.0781359672546387
2.1284103393554688
2.1106467247009277
1.917569875717163
2.0987892150878906
2.10334849357605
2.1629538536071777

mean: 2.0829426765442
```

Figur A.7: Mätning med STE

```
1.8136720657348633
1.7885019779205322
1.6730027198791504
1.604048252105713
1.869687795639038
1.6171824932098389
1.6491599082946777
1.6751024723052979
1.7714695930480957
1.6466538906097412

mean: 1.7108481168747
```

Figur A.8: Mätning med ZCR

```
1.7019670009613037
1.9257009029388428
1.9431686401367188
1.7126250267028809
1.6269760131835938
1.6174628734588623
1.7849853038787842
1.7889037132263184
1.7365400791168213
1.6724770069122314

mean: 1.7510806560516
```

Figur A.9: Mätning med ZCR och STE