Bachelor Thesis

Computer Science and Enginnering, 300 credits

Intelligent chatbot assistant: A study of Natural Language Processing and Artificial Intelligence

Computer Science and Engineering, 15 credits

Halmstad 2020-06-01

Linus Lerjebo, Johannes Hägglund

HALMSTAD UNIVERSITY

Foreword & Acknowledgements

After five months of working on this thesis, we would like to take this moment to thank some people who made this all possible. Firstly we would like to thank our two supervisors Kevin Hernández Diaz and Fernando Alonso-Fernandez. Thank you for the continuous support every week and assistance to come up with new ideas and how to move forward every time we felt lost.

To Viktor Björk at Starbirch thank you for allowing us to come and do this project with you. You have been there and assisting us in every way possible to make sure we can do our job as well as possible. To Ludvig Linse at Narratory, thank you for the coaching and assistance throughout the work and saving us many hours of headache that we would have had to go through without your Narratory. Good luck to you both in the progression of this startup and all the projects to come in the future.

Lastly but most importantly to Erik, our classmate, our collaborator, and most of all, our dear friend. We have gone through so many ups and downs this spring. It's been an incredible emotional roller coaster that we all will remember for a very long time. In the end, we managed to prevail even though all hardships that came during the work and it wouldn't have been possible without you. Also, thank you for taking the heavy load and took it on you to write a separate thesis on your own.

Linus Lerjebo & Johannes Hägglund, Halmstad, April 22, 2020

Abstract

The development and research of Artificial Intelligence have had a recent surge in recent years, which includes the medical field. Despite the new technology and tools available, the staff is still under a heavy workload. The goal of this thesis is to analyze the possibilities of a chatbot whose purpose is to assist the medical staff and provide safety for the patients by guaranteeing that they are being monitored. With the use of technologies such as Artificial Intelligence, Natural Language Processing, and Voice Over Internet Protocol, the chatbot can communicate with the patient. It will work as an assistant for the working staff and provide the information from the calls to the medical staff. With the answers provided from the call, the staff will not be needing to ask routine questions every time and can provide help more quickly. The chatbot is administrated through a web application where administrators can initiate calls and add patients to the database.

Sammanfattning

I samband med utvecklingen av Artificiell Intelligens har sjukvården på senare tid etablerat sig som en ledande målgrupp. Trots utvecklingen av nya avancerade tekniker är de fortfarande under en tung belastning. Målet för detta arbete är att undersöka möjligheterna till en chatbot vars syfte är att lätta belastningen på sjukvårdspersonalen men samtidigt ge en garanti till patienterna att de får den tillsyn och återkoppling som behövs. Med hjälp av diverse tekniker så som Artificell Intelligens, Natural Langugage Processing och Voice Over Internet Protocol kan chatboten kommunicera med patienten. Chatboten fungerar som ett assisterande verktyg för sjukvårdspersonalen så att de kan använda svaren från samtalet agera på ett sätt för hjälpa just den patienten och inte behöva ställa rutinfrågor om igen. Chatboten administreras via en hemsida som kopplar samman de olika komponenterna. Administratörer på hemsidan kan starta samtal och spara ner klienter som skall ringas upp i databasen.

Contents

A

1 Introduction

1.1 Problem description

The medical field is under constant pressure everything form people getting regular sickness and wanting to know what to do about it to people who need surgery. The staff working in the industry has to spend time answering and asking questions that are very much alike each other taking away time from those who need it.

This project was made possible by the company Starbirch. They posted an intriguing idea on creating a call-up chatbot built for medical use. The project description had two stages, researching components that are needed and creating the bot.

This thesis is to investigate how to create a chatbot and then implement it using ML and AI to relieve some of the staff's workloads and allow the patients to get more regular checkups. This contains different sub-processes which should be responsible for managing the different steps in the call. Some of these steps include *Voice over Internet Protocol (VoIP)*, *Text-To-Speech (TTS)*, *Natural Language Processing (NLP)*, *Speech-To-Text (STT)*, *Database storage* and *data analysis*. These are some already established technologies, but the difficult part will be to differentiate which companies provide the best technologies that also fit the project requirements.

There are a lot of sub-processes that need to be researched, so the project has been divided into two different theses. This thesis will be focusing on TTS, STT, NLP, and data analysis of how they work and which company offers the most suitable services. The second thesis[1] focuses on VoIP, database storage, and how a whole system can be made on a theoretical level. In the end, the knowledge of both theses will be used to construct a prototype system.

1.2 Machine Learning in the medical field

Within the last few decades, new AI-technologies have arisen to help the medical industry in all different aspects. Some have been a success, and some others not, but it has made a significant impact in healthcare-industry, so large that some experts think that AI-technology could replace human physicians. Artificial intelligence (AI) is a collection of different technologies, such as artificial neural networks (ANNs), deep learning and natural language processing (NLP), these technologies support various tasks, but they have been of great importance for healthcare since the 1970s [2].

Research for applying machine learning (ML) to predict cancer has been a significant interest. Researchers have been focusing on using methods such as examining the patient-data in an early stage to predict the symptom before it appears. ML can find patterns and relationships from big data sets, which can be extremely helpful, for example, in predicting cancer. In the last years, thanks to machine learning, the accuracy for predicting cancer has significantly improved by 15% - 20% [3].

AI technology does not only show to be helpful in predicting cancer. It can help to improve almost every area, from monitoring the patient to surgery.

1.3 Purpose & Goal

The long-term purpose of this system is to relieve the medical staff from some of their heavy workloads. It will primarily be made to help the medical staff in Sweden due to the language, but it should be easy to expand beyond Sweden with a language switch. While creating security for the patient, this thesis will promote the concept of using AI and ML in the medical field in the future.

This thesis's goal is to research and create a chatbot consisting of many different subsystems and make sure they work together. In the later part, the prototype should be exposed to tests. This is for the company to proceed with working on the functionality. Our thesis will work as the groundwork for the company to see if it holds up in functionality and be worthy for the company to keep working on the idea. Because the groundwork should be as stable as possible, it is essential to research the different subsystems individually to know which components to go for. The second task will be to create an actual bot and start doing some quality tests.

1.4 Requirements

Considering the mentioned purpose and goals, the specification that the thesis will answer are:

- What components are needed for the autonomous call-up process?
- Get a deeper understanding of each component. How do they work?
- Which company offers the best text-to-speech service?
- Which company offers the most accurate speech-to-text service?
- Do they offer trustworthy services in Swedish?
- How is the price compared to the quality?
- Create the dialog and response handler for the prototype.

1.5 Limitations

While many different companies present their cloud computing services to scale down the problem to a reasonable length, the decision was made to focus mainly on Amazon Web Services and Google Cloud Platform. The reason why to focus on these two in specific is that both Amazon and Google are two very well established companies. Both of the cloud computing services have been available for over a decade, are well developed and offer a lot of different services. When developing the prototype, there will not be any data analysis because of a lack of data and time constraints.

2 Background

2.1 Current Technologies

As previously mentioned, TTS, STT, and NLP as stand-alone concepts already exist and are established in some monotonous tasks. This project is to take the technologies one step further and interweave them together to create an entire system, which is to solve a specific task. One example of this type of system integration is *Google Duplex*. This product, developed by Google, is the result of existing building blocks built to simplify the use of virtual AI Google Assistant on Android phones. Currently, available solutions include reservations of restaurants and hairdresser appointments.

Two employees at *Google*, Yaniv Leviathan as Google Duplex lead, and Yossi Matias as an engineering manager on the Duplex project describes *Google Duplex* as an AI system for solving real-world tasks over the phone. They do mention that this system has been one of the long-termed goals for the company, and since it's only capable of restaurant reservations and hairdresser appointments, there are more functions to come. It's essential to provide a pleasant and natural conversation between humans and computers. This includes implementation of *deep neural networks* [4].

Amazon's Alexa is similar technology to Google's virtual assistant but launched by Amazon. Using deep neural networks, Alexa analyses text and speech to create human-to-computer interaction and vice-versa. With Alexa, users can play music, make calls, set routines and appointments, shop and control smart homes [5].

Amazon Echo, launched by Amazon, is a smart speaker connected to Alexa Voice Service, which gives the user opportunity to use the features for Alexa. All they have to do is to ask [6].

2.2 Cloud Computing

Cloud computing continues to grow. It provides benefits for maintaining data and efficient computation of data. **The National Institute of Standards and Technology** [7] defines *Cloud computing* as "A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g networks, servers, storage, applications, and services)". Cloud providers that are open for everyone, and anyone can access their resources, are called *Public clouds*. Clouds that are accessible for a specific organization are called *Private clouds*. Private clouds divide their resources into data-centers so an organization gets what they want and can store private information. *Hybrid clouds* are described as both public and private clouds [8]. Although if cloud computing brings a lot of benefits, there are challenges when integrating with clouds. One of the biggest challenges is security [9], such as data security, network security, data confidentiality, data integrity, authentication, authorization, etc [10].

2.2.1 Service models

Infrastructure as a Service, *Platform as a Service*, *Software as a Service* are the three layers in the *service model*. An organization that's looking for a cloud provider wants clear instructions and a clear overview of the services. The service models give an overview of the available services, and it's up to the organization what model to chose.

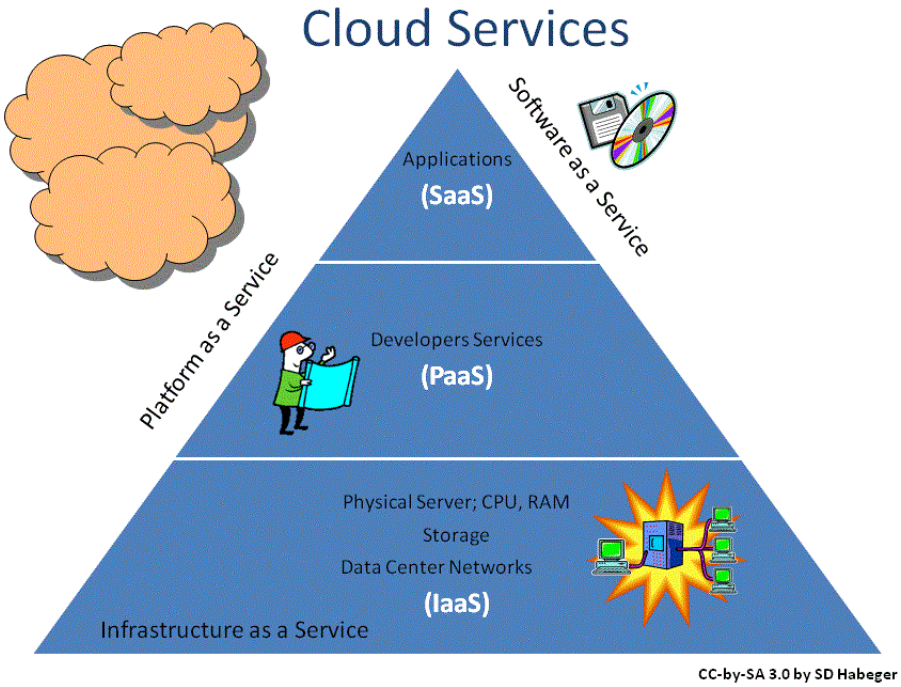


Figure 1: Cloud computing service model, image taken from [11]
.

Infrastructure as a Service (IaaS) [12, 13]: Cloud providers giving this service deliver servers, storage, network and operation systems to users. IaaS is frequently used for hosting websites where users can create, destroy, or manage virtual machines and storage. IaaS can be interpreted as a hosting service that includes network access, routing services, and storage. `Amazon Elastic Compute Cloud`, `Rackspace` and `GoGrid` are such providers.

Platform as a Service (PaaS) [13]: Here cloud providers host everything. The user will pay for monthly usage and can pay for other resources if needed. Access hardware and software tools over the internet, run, build, test, and deploy applications faster and less expensive. Use the PaaS for such as application programming interfaces (APIs) and Internet of things (IoT). Benefits here are similar to IaaS, access to a pool of computing tools, and visualization of hardware.

Software as a Service (SaaS): Software as a service makes it possible for users to run software through the cloud, but only users that pay for the service have access to it [13]. This service does not include any implementations or programming from the user's perspective, but there may be changes and configurations in the software [12]. Created applications are hosted, patched, and upgraded by cloud providers, and the customer can always access via the internet. Also, here some of the benefits are minimal cost and straightforward customization. SaaS clouds can modify applications easily, which results in precisely what an organization is looking for [14].

2.3 Deep Learning

Deep learning is a field in machine learning, invented in the purpose to learn and operate as a human brain does. It is used to identify objects and find deviations in data. Big data-sets called "training-data" used as input for the algorithm, the more the algorithm gets trained, the more accurate it gets [15]. Identify objects from an image, also called image classification is an example where deep learning is applied. People identify objects by practicing and learning from what others say. The algorithm does the same with big data-sets. These data-sets are built on arrays containing pixels that are used to train the algorithm [16]. Deep neural networks can be classified with different networks depending on the problem and approach that's needed. Artificial neural networks (ANNs), convolutional neural networks (CNNs), feedforward neural networks (FNNs) and recurrent neural networks (RNNs) are such networks [15, 16]. The neural network is built on input layers (initial data), hidden layers (computation using weights and producing an output), and output layers (represent the result of computations), which is illustrated in figure 2.

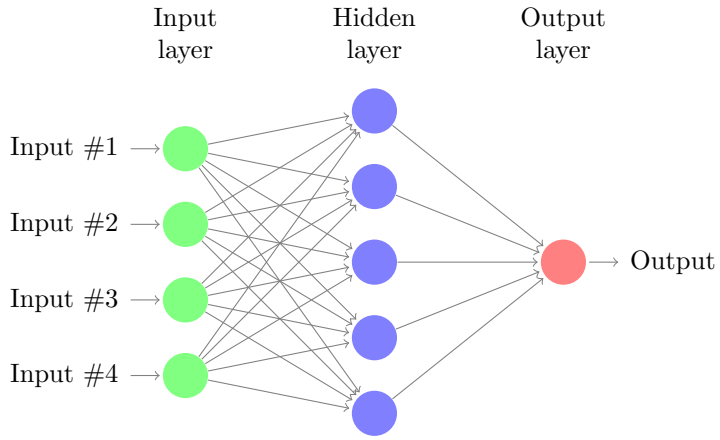


Figure 2: A neural network

Language and speech such as automatic speech recognition (ASR) and NLP are processes where **RNNs** are commonly used [16]. RNNs can predict the next character or word in the input by storing words and characters from previous input. The input contains one element at a time, and previous information is stored in the 'hidden state vector' [16].

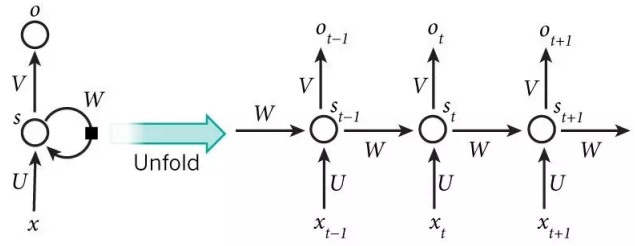


Figure 3: 3-layer Recurrent Neural Network

$$s_t = f(U * x_t + W * s_{t-1}), f(z) = \frac{1}{1 + e^{-z}}, \tag{1}$$

$$o_t = g(V * s_t), g(z) = \frac{e^z}{\sum_k e^{z_k}} \tag{2}$$

The process of unfolding a full RNN is illustrated in figure 3. Revealing the network means write it out for the complete sequence of words, which means that figure 3 is a full 3-layer network for a sentence of 3 words (one layer for each word). $x_t = (x_1..., x_T)$ represents the input sequence at time step t, $s_t = (s_1...., s_T)$ represents the hidden state vector at time step t which is the memory for the network. The calculation for s_t depends on the previous input (x_{t-1}) and the current input.
$o_t = (o_1..., o_T)$ represents the output at step t which is a vector containing probabilities for an certain word [17]. W, U, V also called weight-parameters/weight-matrices are the parameters included during computation [17, 18].

CNN, also called ConvNet, is commonly applied for image processing such as detection, segmentation, and recognition. The input data comes in multiple arrays, and the dimensions can differ from 1D to 3D depending on if the input is a signal, image, or video. The image below illustrates the process of labeling an image. Input data represents a 2D array containing pixels from the picture, and CNN detects objects, people, animals, etc. Then feeding the information to the RNN gives an output that represents a description of the image [16].

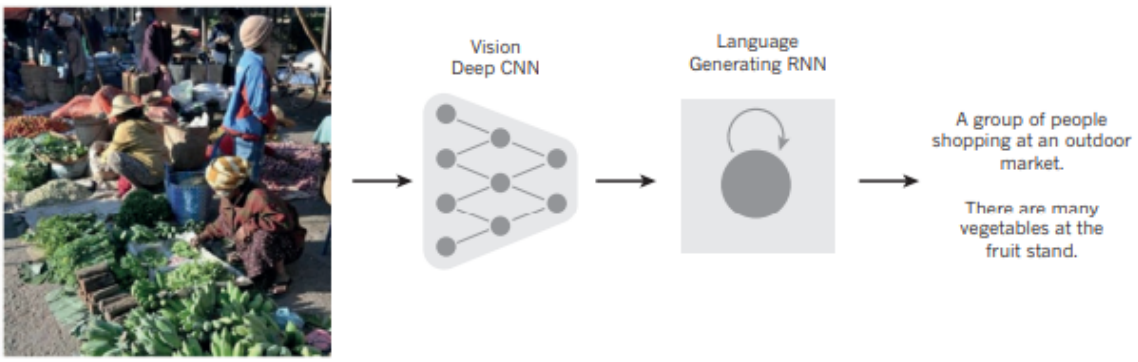


Figure 4: From image to text using CNNs and RNNs

2.4 Natural Language Processing

In recent years NLP has been thriving more than ever due to the amount of data available and increased amount of computational power. It has been especially successful in the healthcare industry by predicting diseases based on health records [19]. NLP is an area of application in artificial intelligence, and it's used to analyze a large amount of text-data, voice-data, and language-data. This data is created and extracted from interactions between humans and computers. Natural Language processing is built to perform four different tasks, part-of-speech tagging (POS), chunking, named entity recognition (NER), and semantic role labeling (SRL) [20].

2.4.1 Part-of-speech tagging

In POS-tagging, the computer takes all the words from a text and sorts them into different syntactic roles that can be a noun, adjective, or singular. Some words may have multiple roles or different roles, depending on the context. To sort all the words, most system uses two different techniques, *rule-based*, *Stochastic/machine learning* and some systems use both [21]. Both of the methods are valid, but machine learning/stochastic are considered stronger because they are trained automatically and can, therefore, be expanded easier.

Rule-based tagging is made from getting a set of rules taken from a dictionary or lexicon. In the first run, the machine set the tags of the words according to dictionary rules. Then it runs through the words that have multiple potential tags and changes them according to handwritten rules concerning the words before and after the incorrect tag instead of how the dictionary says [22].

Stochastic methods are using a machine that trains on a corpus text to be able to pick the most probable tag for each word, often by a hidden Markov model (HMM). The HMMs goal is to predict the future given the present without the need for knowing the past [23]. The HMM is built on having a set of observations and possible states, with the words being the observations and their hidden state is the POS tag. By calculating the transition probabilities and emission probabilities and take the highest value, the answer will become the most probable tag. Transition probability estimates the likelihood of the current word having a tag given the previous tag. Emission probability gives the chance of a specific word being a particular tag [24].

2.4.2 Chunking

Chunking works by taking in POS-tags but returns chunks of words as a phrase. Different phrases are depending on the content of the phrase for example, it could be a noun phrase, verb phrase, or prepositional phrase. To define a phrase, the program looks at the POS-tagging rules and splits the sentence depending on the previous and upcoming words [25]. This is especially useful when wanting to extract information from the text. Still, some words can have many different POS-tags or words that usually mean one thing but are separated. One example of this is “South africa” which should be tagged as the name of a country not “south” as in the direction and “Africa” the continent [26].

2.4.3 NER & SRL

NER works very similar to chunking, but instead of dividing it into grammatical pieces, it extracts the most important entities such as names, places, and times. The second subtask SRL labels the words in a sentence, some typical roles that are used are Agent, Patient, and Location. The agent is the active word that does something, and the patient is the role that the agent is acting upon [27]. These two are quite alike but are used in different scenarios. NER extraction allows the computer to know who was involved and where they were, for example. While SRL only assigns semantic roles, which means two different phrased sentences that have the same meaning will get the same grammatical purposes.

2.5 Text-To-Speech

Every TTS-system has four standard components [28]:

- text analysis and normalization.
- phonetic analysis.
- prosodic modeling.
- speech synthesis.

By analyzing the input, the machine organizes the text to a list that contains numbers, abbreviations, acronyms, and idiomatics so that the normalization can transform the text into readable content. The text normalization checks if the input is in full-length text form if it is a number, acronym, or just not in text form. The normalization goes to the list that was previously mentioned to transform the word into text form. The other primary purpose of the normalization is to identify the punctuation and pauses on the input.

Phonetic analysis, the second step, takes the normalized text and converts the orthographic text to phonetic text. Phonetic texts are based on the phonetic alphabet, which is of the regular alphabet, but how the letters would be pronounced aloud. The machine has two different ways to know how to pronounce the words correctly. The first method it can use is to have a database with a dictionary that has the correct pronunciations in them. This method is rapid and has good pronunciation, but it needs an extensive database to store the words and breaks down if the word is not found. The second method the machine works based on the rules of the language. The main advantage of this method is that it does not need an extensive database and will not break down, but it may lack quality in cases where the word is pronounced differently to how the rules state [29].

Now that the pronunciation is ready to go, the third step to make a synthetic voice is adding the human elements such as tone, rhythm, and emotions, also known as the prosodic modeling. By changing between a high and low pitch at the end of the utterance, you can indicate if it is a question or a written sentence. The pitch change makes the listener apprehend what kind of emotion the AI is talking through to make it seem more realistic, for example, angry people usually have a broad pitch range while depressed people have minimal pitch ranges [30].

The speech can be generated sounding standard or neural. The standard is generated using two different techniques called *Concatentive TTS* and *Parametric TTS*. The neural voice is generated by combining WaveNet with one of the standard methods.

2.5.1 Standard voice

Standard TTS generates using one of the mentioned techniques, concatenative TTS or parametric TTS. Due to these two techniques, the speech lacks emotions, feelings, etc. Which results in more emotionless expression.

Concatentive TTS is based on a huge database. The database contains short segments from an already recorded speech. During the process, the input data is compared with database-data (also called units) to generate a complete word, phrase, or syllable [31]. The input is a text, and the first stage for the speech synthesis is to transform the input text to a target-specification. This target includes information such as phonemes (how to synthesize the text) and prosodies features such as pitch, power, and duration.

Statistical Parametric TTS, parametric because the speech is described with different parameters and statistical because it uses statistics to analyze these parameters. This model consists of multiple parameters that are fed into a synthesizer. The synthesizer is combined with two modules, first module for producing the voice and second for continuously measure and modify values of vowels, pronouncing, etc. Parametric systems offer control over such as pitch level, loudness, duration, and parameters are updated every 10 ms [32].

2.5.2 Neural voice

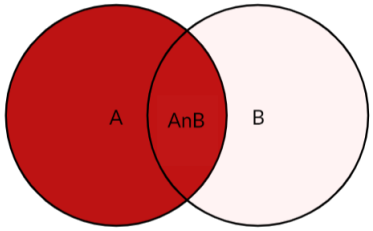
This voice is more deeply analyzed and modified, and by applying a deep neural network called WaveNet will generate a more 'human-like' speech. **WaveNet** is a deep neural network which is a

follow-up to that technology and is used by Google's [33] systems to create authentic TTS. WaveNet is a technique, combined with concatenative TTS, which generates raw audio through the technology of neural networks. The main component of WaveNet is causal convolutions, which means that the model cannot depend on any future timesteps (steps between layers in the neural network). WaveNet uses this concept to create many samples of audio and make each sample dependent on the previous timesteps it takes from the network. There are different methods of guiding this model and it takes form on conditional probabilities (3) where $\mathbf{x} = (x_1 ...., x_T)$ represents samples from the audio waveform from t = 1 to T;

$$p(x) = \prod_{t=1}^{T} P(x_t | x_1 ... x_{t-1}) \tag{3}$$

$$P(x_t | x_{t-1}) = \frac{P(x_t \cap x_{t-1})}{P(x_{t-1})} \tag{4}$$

Equation (4) is called conditionally probability and can be expressed as 'What is the probability that x_t occurs given x_{t-1}'.



$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Figure 5: Visualization of conditionally probability

2.6 Speech-To-Text

The most common and most straightforward way for humans to communicate with each other is through voice. Speech to text (STT), also called speech recognition (SR), is the most commonly used technology for creating interactions between humans and computers. This technology does not only improve and enables human-computer communications, but studies have also shown improvement and efficiency for human-human communications [34]. Suppose a Chinese travels to Sweden, and this person is not familiar with the English language. Here the most efficient way for this person to communicate with swedes is to translate sentences from Chinese to either English or Swedish, and this can be accomplished just by speaking through the microphone to produce the translated text. Supported applications are voice search, personal digital assistant, gaming, smart-homes, embedded systems within cars etc [34]. The architecture of the SR includes four components [35], signal processing, feature extraction, language model, and classification.

2.6.1 Signal processing

This stage is the core of the process, and here the speech is recorded using a specific sampling frequency, e.g. 16kHz [36]. The interesting information in the waveform is actually extracted from a short term

amplitude spectrum and by transforming this spectrum to a representation of parameters that allows for analysis of phonemes in the speech [35].

2.6.2 Feature extraction

In this stage, the parametric representation of the signal (from the first stage) is represented as a vector by compressing the input signal [37]. There are many techniques for feature extraction, but one powerful and commonly used is Mel-frequency cepstral coefficients (MFCC) [38]. The purpose of MFCC is to analyze and transform frames within 25 ms to imitate and reflect human speech [35].

2.6.3 Language model

This is previous knowledge about the language such as grammar, word combinations, pronounces, linguistic sounds etc [35]. The language calculates probabilities for each word using the likelihood function (next word depends on previous words). Suppose a sequence "Hello World", there is a higher probability for the next word "World" to occur than "XYZ" since "World" is grammatically correct.

2.6.4 Classification stage

The last step before producing an output of words is the classification stage. Here a combination of language model and feature extraction is used to recognize the words in the speech. For the computation, three methods can be used, HMM, Artificial Neural Networks, or Support Vector Machines [35].

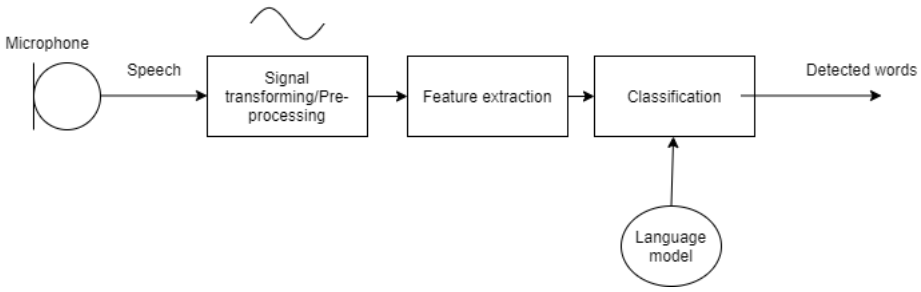


Figure 6: Visualisation of an ASR system

3 Method

3.1 Task Specification

This project focuses on an early stage idea and includes testing and research to see which methods are viable. Due to this, the workflow has followed an agile development model. With the project being a proof-of-concept, new concepts and techniques are bound to arise while researching. That resulted in a need to study those new concepts to be able to understand the problem correctly. The agile development model works well since the project involves a lot of testing and reviewing of which parts satisfied the requirements.

With the project, the agile model decided it was time to find what components most autonomous call-up systems use by researching similar technologies that exist, determine what parts our system could use and which are unnecessary. To be able to get a deeper understanding of how each component works arose the need to read journals and articles on how each component is built. The next step to take is to look at what companies offer these components and test them against each other to see what would suit this project the best.

The choice of cloud provider was made by comparing two candidates, Google Cloud Platform (GCP) and Amazon Web Services (AWS). Before starting comparing the two services objectively, some API:s were tested to make sure that it would be possible to integrate the prototype with whichever deems more suitable. The API:s that were tested was Amazon Polly, Amazon Transcribe, Google Cloud TTS, and Google Cloud STT. Due to previous knowledge, the test coding was done in Python, but they both work with more languages, for example, Java and Golang, to name a few.

3.2 Availability

The chosen provider was based on available services that are attractive for this project, what languages the different speech synthesis supports, performance testing, benefits, drawbacks, and pricing.

3.2.1 Google Cloud Platform

GCP is one of the most significant and most growing cloud providers today. GCP provides over 90 products of different cloud computing services that developers can use. Few of them are computing, storage & databases, cloud AI, networking, and big data. Users can sign up for free first 12 months and start using available services, but depending on the usage of each service, the pricing will be different [39, 40].

TTS is one of the cloud services GCP provides, and this service supports 180 different languages, 30+ voices, and few of them are Swedish, English, Danish, and French [41]. One drawback is that the Swedish standard and neural voice sounds very similar, but users are allowed to change pitch, volume gain, and speak rate tuning, which can be customized to produce a more neural voice. To integrate this API with a system, it requires an API key from Google. The API takes input as text to then generate an output representing the speech.

Table 1: Google TTS pricing

Voices/Account type	Free tier	Paid usage
Standard voices	0-4 million characters	$4 USD/1 million characters
WaveNet voices	0-1 million characters	$16 USD/1 million characters

STT is another available service which enables developers to convert speech to text from 120 different languages. STT can be integrated with four different models, which are gathered together in two models called standard and premium. The standard models are command & search (voice commands and voice search) and default. The premium models are best for phone calls and audio from video. The developer can either choose if the audio-data will be stored or not stored. By storing new data, Google algorithms can be improved. Few supported languages for STT are English, Spanish, Swedish and French. The SST takes a speech as input and outputs a transcribed text from the speech. One drawback is that premium models can only be applied in certain countries, and Sweden is not included. Implementing this API for phone calls without specifying the premium model could cause loss of information in the conversion, and only parts of the speech will come along in the text. For integration with the API, also here it requires an API key from Google. This service is free for files no longer than 60 minutes otherwise, it is priced per 15 seconds [42].

Table 2: Google STT pricing for duration > 60 min

Data Logging/Models	Standard models	Premium models
Without data logging	$0.006/15 sec	$0.009/15 sec
With data logging	$0.004/15 sec	$0.006/15 sec

3.2.2 Amazon Web Services

AWS is also a cloud computing platform providing services within computing, networking, storage, databases, analytics, application services, and mobile applications. Also, AWS offers twelve months of free tier and user pay-per-use for each service [43].

Amazon Polly is a service provided by AWS. This service is capable of converting text into speech for either standard or neural voice. Polly supports conversion for multiple languages such as Swedish, English, Danish, French, and Chinese. Polly can generate different sounds for producing the speech, either female or male voice. Few examples of voices in Polly are Nicole and Russell (English), Astrid (Swedish), and Enrique (Spanish). Polly supports both standard and neural voices, but neural is restricted to four voices (English, British English, Brazilian Portuguese, and Spanish). Integrating this API within a system requires credential keys (access and secret keys) [44].

Table 3: Amazon Polly Pricing [45]

Voices/Account type	Free tier	Pricing
Standard voices	0-5 million characters	$4 USD/1 million characters
Neural voices	0-1 million characters	$16 USD/1 million characters

Amazon Transcribe is another service from AWS, which converts speech to text using machine learning. The input is an audio file or microphone, and the output will represent the transcribed text from the input speech. Transcribe can be used when analyzing customer calls or produce real-time subtitles. Transcribe can identify speakers in an audio file called speaker identification. Another service, extended version of Transcribe, is Amazon Transcribe Medical, which is an ASR API service in purpose to support medical applications. Since the output represents a JSON, it is very efficient when it wants to apply analyzing techniques on the data for medicinal purposes. One drawback of both services is that they do not support conversion for Swedish speakers. Also, this service requires credentials for high-security [46]. The pricing for this service differs from what region that is chosen, the chosen region for table 4 is EU (Ireland).

Table 4: Amazon Transcribe Pricing [47]

Free tier	Pricing
audio length<60	$0.0004/second

Table 5: Google vs Amazon

GCP	**AWS**
The developer has the opportunity to optimize the system for either long or short speech	Transcribe has fewer accents for the English language. But supports six other, Arabic, Chinese, French, German, Portuguese and Spanish. Transcript does not support Swedish speakers.
Can be implemented using either Python, Node.js, Java, C++, C, PHP and Ruby.	The input of any length, no specific option for optimization.
Data logging is optional. The developer can specify if speech gets logged or not.	.NET, Go, Java, Javascript, PHP, Python, and Ruby.
Not allowing developers to customize and add new vocabulary.	Store the voice data for further improvement of the machine learning algorithm. Though deletion of recording is an option.
API key or bind a key for the project that can be used for all products	Allows to create new vocabulary but only supported for English.
Supports both standard and neural voices	To integrate with API:s within a system, secret and access key is needed
TTS API with over 180+ languages and 30+ voices.	Supports both standard and neural voices
Google Security Model, end-to-end process to focus on customer's security. Includes security team, trusted server bots, data encryption, etc.	Amazon Polly (TTS) supports 28 languages with 57 different voices.
	Users can control where data is stored, administrate the access control, detect frauds, logging and monitoring the services.

3.3 Performance

Performance testing is an essential part of the election, and tests such as quality for TTS systems and accuracy for speech recognition will be studied. To get an accurate speech comparison between GCP and AWS, it requires a lot of data, and due to the time and cost constraints, previous papers will be used.

The accuracy test for speech recognition services will be done using a very common approach called Word Error Rate (WER). WER is a methodology for measuring the accuracy of a speech service. A person speaks through a microphone, the speech is used as input, and the STT produces the output. The accuracy for the speech gets computed using equation (5), and to obtain a significant and trustworthy result, at least two hours of data are required, which is very time consuming and expensive [48].

$$WER = \frac{I + D + S}{N} \times 100 \tag{5}$$

The WER will be calculated by comparing the original text that the speaker read and the transcribed text from the STT service.
S - represents a number of substitutions, words that got exchanged by other words as Hello got exchanged by Hey.
I - represents a number of insertions, new words that got inserted.
D - represents a number of deletions, words that got deleted.
N - represents the length of the original text.
Summing all errors, dividing with the length and multiplying by hundred will give the error rate for the speech [48].

A company named Rev [49], in 2019, launched its transcription service that is capable of converting audio and video files into text. Similar to other STT services, this service is also built on ASR and NLP. Along with GCP, AWS, and Microsoft Azure, they decided to calculate the accuracy for each speech service and compare the results with their service. The test included a collection of 20 different podcast episodes, and each episode could range to 18 hours of audio. Rev also published all sample data from the performance test which could be extracted from their Google Drive. Each sample is shown in Figure 7, and the mean WER and the standard deviation for each provider is shown in figure 8, since the decision was to compare GCP and AWS, these are the only providers compared in the figures.

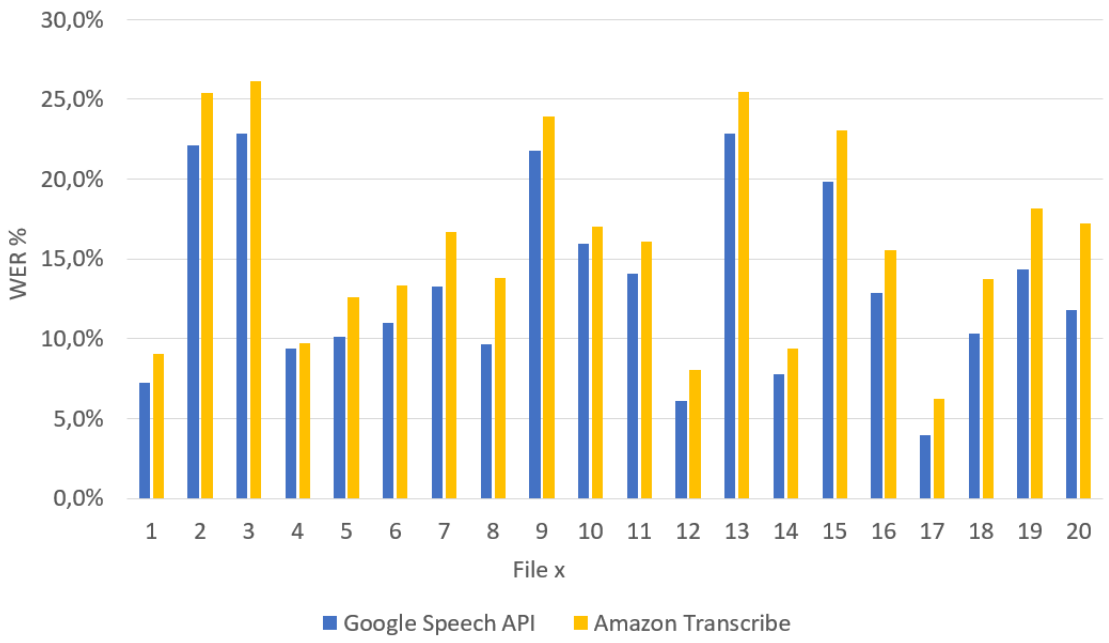
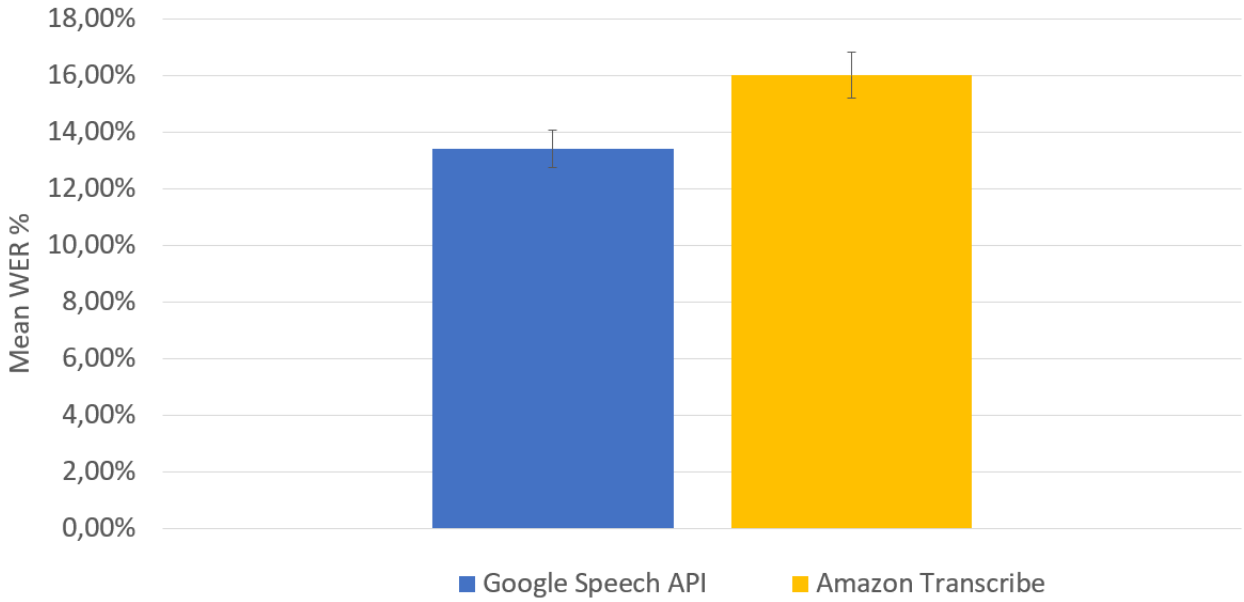


Figure 7: WER for each sample for STT performance



$\mu_{\text{Google}} \approx 13.4\%$ $\sigma_{\text{Google}} \approx 5.8\%$
$\mu_{\text{Amazon}} \approx 16.0\%$ $\sigma_{\text{Amazon}} \approx 6.0\%$

Figure 8: WER (mean and standard deviation) for STT performance

The most common way of measuring quality performance for TTS systems is by letting humans inter-

act with these systems and give their opinion about the experience. The methodology is called Mean Opinion Score (MOS) and is the average score of the participant's opinion [50]. The process involves asking participants about their experience and let them rate it from 1 - 5 [51].

Table 6: Ratings included in MOS [51]

Rating	Quality
1	Bad
2	Poor
3	Fair
4	Good
5	Excellent

A previous paper compared the experience and congeniality for TTS systems at different providers, and two of these providers were AWS and Google Cloud. The comparison was made by selecting 18 TTS voices and three human voices. Both US-based male and female voices were chosen [52], and for the human voices, members from the research team were selected. For the study, a survey was created which contained questions about the audio and the experience. The inquiry was answered and completed by 1090 people, where 47 participants chose to analyze Google C (Female), 51 Google A (Male), 46 Polly Matthew (Male), 52 Polly Sally (Female), and 50 Polly Johanna (Female) [52]. During the survey, participants listened to an audio clip containing 909 words, then rate the experience, and the last answer some questions about the audio clip to test the clarity of the voice. The result of the survey is shown in figure 7 and demonstrates the MOS score, median score of repeating the audio (0-10), the mean clarity between 0-100, and mean grade 0-10. For Google's voice, participants found C sound 43% good and 17% excellent and for voice A 37% good and 10% excellent. For Amazon Polly, Joanna (36% good and 14% excellent), Matthew (36% good and 14% excellent), and Sally (39% good and 15% excellent).

Table 7: Results taken from the survey [52]

Voice	MOS	sd	Listen Again	Clarity	sd	Grade	std
Google C	3.7	0.9	7	55	39	4.3	2.6
Google A	3.4	0.9	4	47	39	5.1	3.0
Polly Matthew	3.6	0.9	5	43	42	4.5	2.7
Polly Sally	3.5	0.9	4	52	42	5.4	2.6
Polly Johanna	3.4	1.0	4	51	42	5.0	2.7

3.4 Prototype

While comparing AWS and GCP in the availability section, it is clear that both providers offer similar services for close to identical prices. During the performance section, GCP outclasses AWS by small margins. Based on the performance tests and the fact the Amazon Transcribe does not support translation for Swedish, the prototype will be done using Google services. The company also preferred to use Google services because they have been using it before. They have also implemented their API for dialog building built on Google Services called Narratory. Narratory is built upon Dialogflow, which made by Google to let the computer understand what a human says using NLP during a computer to

human conversation. While using the company API, it becomes possible to have a more fluent dialogue with the patient. This is due to how much training Dialogflow already has.

3.4.1 Dialogflow

Google's machine learning powers Dialogflow to let a user interact with an AI in different ways. The communication works in both voice calls, chatbots, and personal assistants such as Siri, Alexa, and Google Assistant. The AI is called an agent that can be given commands to specify how it should work. Some commands decide what kind of answers the agent should search for and which words have the same intentions [53].

To store specific and interesting parameters from the call, webhook-request is used. Webhook is HTTP callbacks that are triggered by certain events in the application [54]. When the event gets triggered, it will either send POST or GET request. POST indicates that the request wants to upload the transmitted data somewhere, usually to a database. GET means that the request wants to fetch the data from the URL. Dialogflow makes a POST request with parameters such as responseId, querytext, webhook-latency, intent, etc [55]. One interested parameter that can be used for analyzing is *intentDetectionConfidence*. *IntentDetectionConfidence* means, how confident the bot is about what it heard and how confident the response matches the correct intent for the dialog. *Webhook-latency*, how long it took for the request to post the data and, in response, get the status of the request.

3.4.2 Narratory

Narratory is used to control and administrate the call flow of both the user and AI in the voice or chatbot application. Narratory is written in Typescript because it is a dynamic language and is suitable for declarative coding. The narrative is built on the conversations taking turns, and when a message is incoming, there have to be some predetermined phrases the AI can look for, and when finding them, the conversation will jump to that segment. The predetermined phrases are called intents short for intention since many words can have the same meaning [56]. By taking in the user intents, the dialog can be more flexible in how to answer. An example of how this work can be seen in Appendix A.2 the narrative can become very complicated and dynamic, especially when having a phrase that can be said in multiple different ways to make it less repetitive to the user. In Appendix A.3 are two examples of how intents work, the AI searches the answer to see if it contains any of the phrases or parts of the phrases.

Also, a webhook is used to store specific parameters from the narratory-dialog, and the developer can choose to store data such as only fallback, none data, or all data from the conversation. The dialog will send POST requests with parameters such as sessionId, agentName, platform, turn, lastTurn, and text [57], which will be uploaded to a database. The webhook URL (logwebhook) and what data (logLevel) to store can be seen in Appendix A.4. The request comes with sessionId and the turn where the sessionId is used to update the database document with the current turn.

Beyond webhook requests, API-request will also be used to fetch data from an URL. The difference between API-request and webhook-request is that API does not trigger on events. An excellent way to enter the conversation could be to greet the end-user with their name. This will be done by fetching the name from a database and send it to the requesting source, which in this case will be the dialog. In narratory, API-requests can be done by defining the bot as "DynamicBotTurn" instead of "BotTurn" [58].

3.5 Related work

One similar to this work, conducted by B.Rystedt and M.Zdybek [59]. They investigated developing a conversational agent in purpose to assist throughout the cooking process. The system uses a conversational agent to search for what recipe to look for. The system is constructed using techniques such as TTS, STT, NLP, DialogFlow, web search, and web scraping. The conversational agent is implemented using DialogFlow, where intents and entities are focused on recipes. Web search and web scraping work as a back-end process that is implemented with Python. STT is being used with Google Speech Recognition, and TTS uses the pyttsx3 module, both built using Python. Ten people made the testing, where each person would try out the features and rate related to expected features.

The feature expectations for this kitchen-assistant went positive. Some persons said that the program had expected features, and others said that there was more than expected. Though there were some feature improvements, some people claimed that the program was a bit too slow, where most issues could be found with STT and TTS. There could also be improvements with the NLP service, where sometimes the bot could not distinguish between 'two', 'to' and 'three', 'tree'. For the TTS, most said that it sounded 'robot-like', also there was some lousy pronunciation of some words.

4 Result

4.1 Prototype

The finished product is supposed to be an autonomous call-up system to help relieve the medical staff's workload by doing regular checkups and allowing users to check and regulate the time of their meetings. In order to test the different functionalities, the prototype will be a checkup for the current pandemic virus COVID-19 to ensure people follow their country's health regulations.

4.1.1 Agent

The agent that was created for this project can be found in Appendix A.4. Some commands, such as agentname and language, are self-explanatory. GoogleCredentials and narratoryKey are used to connect Google Cloud and Narratory to the agent to be able to use their APIs and functionalities. By using logWebhook, previous conversations can be stored to analyze the answers given by users. The calling narrative allows the agent to see what is in the narrative class, which contains the structure of the conversation. UserInitiatives work very much the same, but instead of holding the conversation, it contains questions and phrases that the user might say that diverge from the original narrative. A bridge phrase is used after a userInitiative has been activated and answered, and the agent wants to move back to the original narrative to make it sound more natural instead of just jumping back. The agent's chosen language is Swedish, using the voice "sv-SE-Wavenet-Ad" with a speaking rate of 0.9.

4.1.2 Natural Language Understanding

To understand intents as the ones in Appendix A.3, the agent uses Natural Language Understanding, which is a subcategory in NLP. By having predetermined words and phrases the developer adds, the agent gets trained using those so that it can later recognize if a phrase or part of its in the answer. By adding more words that have the same meaning, the probability that the bot manages to understand correctly gets higher since it has more data to train on.

In Narratory, some intents are defined beforehand and do not have to be trained again. One of these is used at the end of every question asked, and it is called "ANYTHINGd". It catches anything if none of the other intents could find it. This prototype uses this to repeat the question one more time if the user gives a non-valid answer.

4.1.3 Narrative

The narrative that the agent follows can be seen in Figure 9. In the flowchart, the blue and red connectors specify the two main directions the call can take. If the user ends up in the red path, that means something is wrong with them, and they should be contacted by a real person as soon as possible. While the blue path implies that everything is good and the user will just be added back into the database and called again a couple of days later. The black connectors are the paths that can be crossed independently of the previous answers and do not mean that the user needs help.

Depending on the answer on the first "How are you doing?" question, the other questions will be asked in a different order. This is because if the agent sees that something is wrong with the patient, it is more important to ask if they need help rather then if they have met someone. However, even if the user is healthy, they might need help with food or medicine to stay healthy. The blue blocks are statements coming from the bot. The orange ones are questions asked by the bot, and green ones are answers the bot is looking for from the user.

The way to get the dynamic narrative where the questions are asked in a different order depending on the answer is to use the Narratory command goto. Goto makes so the narrative jumps out

of predetermined order. It jumps to a question that has the same label as the string after the goto command.

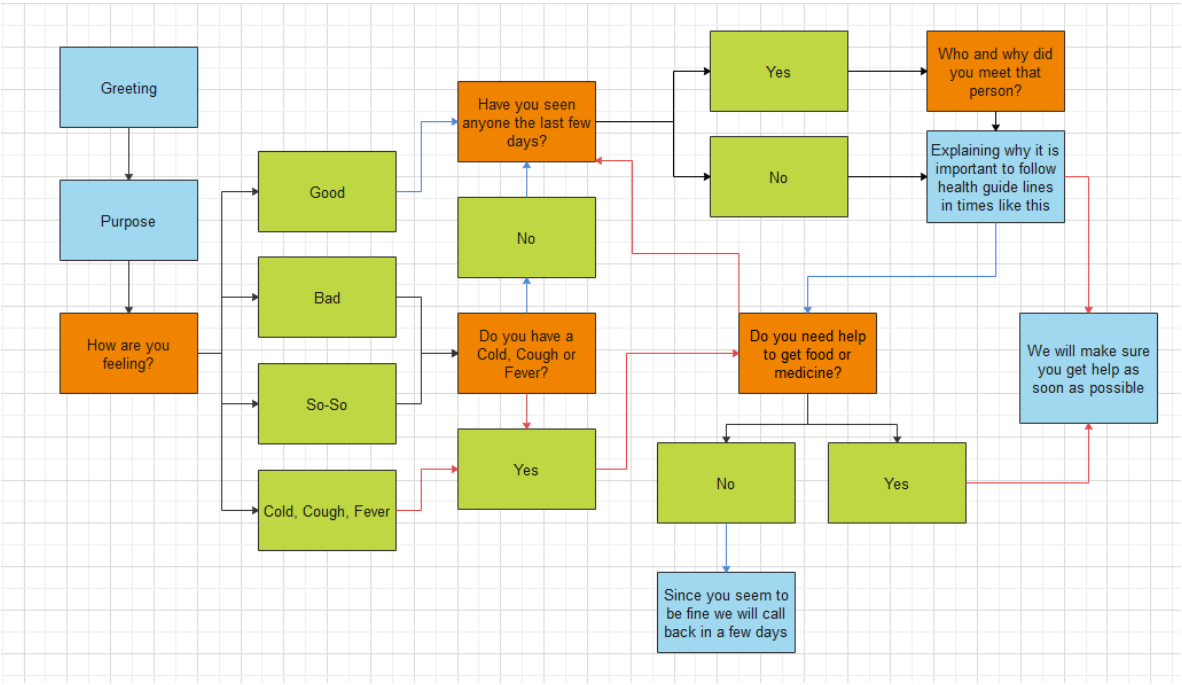


Figure 9: Flowchart of the narrative in the prototype

The dialog in Figure 9 is the final dialog the prototype uses. In the beginning, the conversation was the same narrative every time. Still, it was improved after discussing with a consultant of the company, who had previous experience with Dialogflow and chatbots. The switch-up in the order the bot asks the questions together with being able to phrase the same question in multiple ways, as seen in Appendix A.2 makes it so that even if it is only one narrative, the call will be different every time. This is important because it makes it feel less repetitive for the user to get the most genuine answers. If the narrative is too similar every time, the user might get bored and give specific answers out of habit instead of what they feel.

4.2 Prototype testing

4.2.1 Log

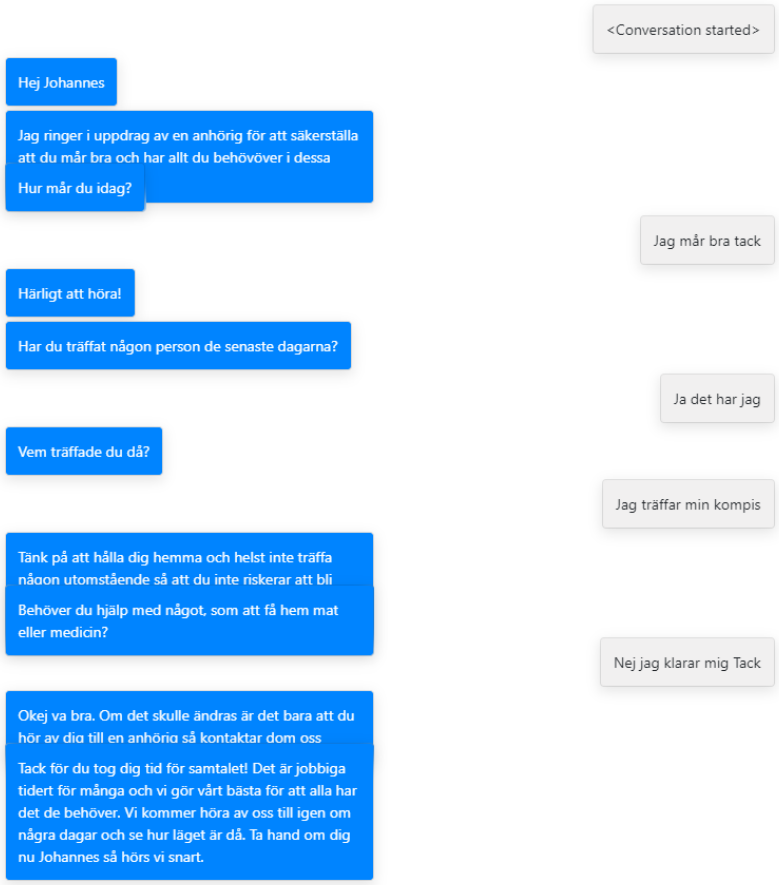


Figure 10: Log from a call with a collaborator

While the call and conversation are active, the database will be updated with new turns as new events occur. Figure 10 illustrates the whole conversation with a collaborator that received a call. The bot-turns are the blue ones, and the end-user response is the white ones. The bot starts the conversation by greeting the end-user with their name. The process of fetching a name from the database can be seen in Appendix A.5. And how the request from the dialog is made can be found under the turn "init" in Appendix A.2. The bot will send request from the dialog to the given URL with data that includes the current sessionId for the session and declared parameters. The URL will take care of the data, use the sessionId to fetch the name from the database, and then assign the transmitted parameter with the name. The response will be in JSON format and include the assigned variable so the bot can make use of it. The bot will explain the purpose of the call and ask the first question, which is "How are you?". As shown in Figure 9, the end-user can take four different paths on the first question. In this log, the end-user had chosen to answer with "I am good, thanks", which goes under the intent feeling good. Therefore the bot decided not to ask about symptoms instead of advice about how to stay away from COVID-19.

4.2.2 Performance

The first testings were to check for any significant flaws that could break the whole prototype that had not been tested during the work process. The first potential flaw in the completed prototype was to check how the system would respond when calling multiple people at the same time. This test was done by calling up two people at the same time, and the bot could handle this without any problem, both phone conversations worked as expected. The second test was to see how the bot would handle the conversation if it did not get any response. The VoIP, which connects the call to the user, has a built-in time limit on each answer; therefore, the agent kept on going to the next question once the time limit went out.

The performance test was done by measuring the intent-Detection-Confidence for each call. The raw data is represented in figure 12 that can be found in the appendix section. Average and standard deviation are represented in Figure 11 on top of each staple. In this experiment, five people were called in three different environments with a pre-defined script, a total of 15 calls, and during each call, the user answered five questions. The environments explained more in detail below.

- **Ideal environment** - In an ideal environment, the end-user gave short answers without any background noise. The result of short answers can be seen from the blue bars.
- **Long answers** - Here, the responses from the end-user were longer, and as in ideal, no background noise applied. The result can be seen from the orange bars.
- **Background noise** - In the third test ideal answers were used but with a background noise of 60-70 dB applied. The result can be seen in the in the gray bars.

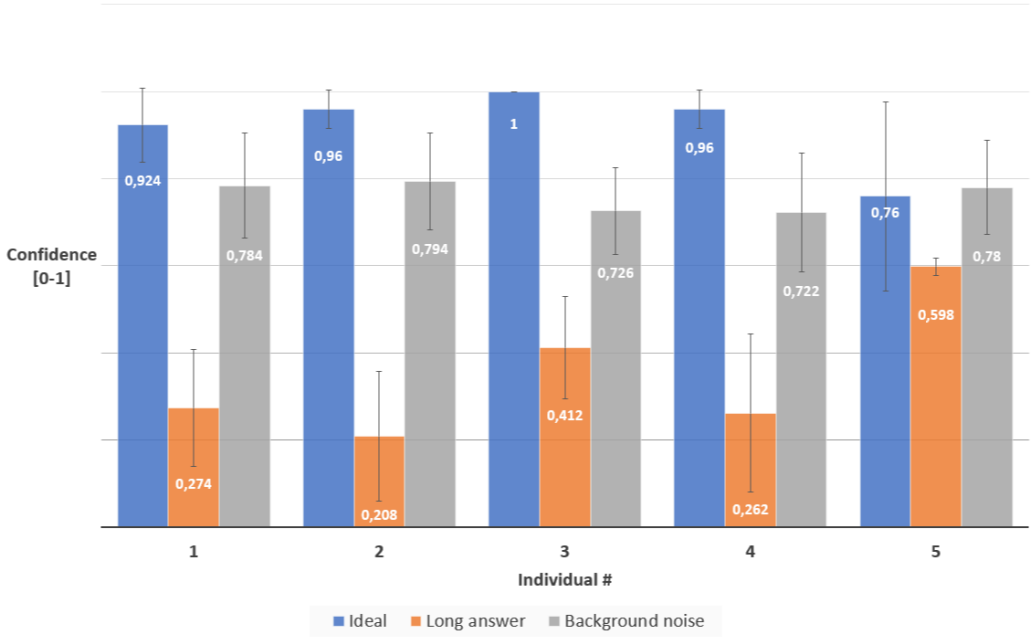


Figure 11: Intent Detection Confidence

The first two tests were made to see both ends of the spectrum when the bot performs best and worst. As seen in Figure 11, the difference is significant mostly because the prototype is still in the early stages and needs to be provided with more data to break down sentences correctly.

The background noise that was used during testing was: other people having a conversation, playing music, and standing next to a road. This test was specifically made to see how good it is at differentiating between who is talking to the phone. The decision was made to only use background noises for short answers because it was clear that the difference between ideal and background sound was small. It is already clear from the picture that the bot handles long responses poorly, and background noise would not make that any better and only state known fact, which is that it can not handle complicated answers.

Each bar represents the average confidence for a specific call. The average was calculated by adding intent confidence for each answer. The total average confidence for the ideal answers is $\mu_{\text{ideal}} \approx 92\%$ with a standard deviation $\sigma_{\text{ideal}} \approx 9.3\%$. This shows, in an ideal environment, the bot can recognize what intent the end-user is relating to in $\mu_{\text{ideal}} - \sigma_{\text{ideal}} \approx 82.7\%$ to max 100% in 95% of the cases.

With long answers, the total average is $\mu_{\text{long}} \approx 35\%$ with standard deviation $\sigma_{\text{ideal}} \approx 3.4\%$. This shows, with long answers, that the bot can recognize what intent the end-user is relating to from $\mu_{\text{long}} - \sigma_{\text{long}} \approx 31.6\%$ to max $\mu_{\text{long}} + \sigma_{\text{long}} \approx 38.4\%$ in 95% of the cases.

With background noise, total average is $\mu_{\text{noise}} \approx 76\%$ and standard deviation $\sigma_{\text{noise}} \approx 3.4\%$. The bot could recognize what intent the answer belongs with from $\mu_{\text{noise}} - \sigma_{\text{noise}} \approx 72.6\%$ to max $\mu_{\text{noise}} + \sigma_{\text{noise}} \approx 79.5\%$ in 95% of the cases.

5 Discussion

5.1 Result

The greatest strength of our prototype is that it can hold a dynamic conversation and answer differently depending on the user's answer. This is explained more in detail in Section 4.1.3. This strength comes mostly from the integration between Dialogflow and Narratory to be able to train the agent with Google's data and use the flexibility of Narratory to phrase questions and sentences in different ways. In Figure 11 that can be found in Section 4.2.2, we can see that the bot handles short answers much better than when the user gives long sentences for answers. Not being able to handle long answers could become a considerable weakness. To counteract this the questions are created with the intent of forcing the user to use short answers. In the same figure, another strength can be found that the bot handles background noise almost as good as with no sound. To be able to filter out background noise and buzzing is essential since it makes so that the user can be outdoors, and the bot can still pick up what is said.

One of the significant drawbacks of the prototype is the language barrier, where the Swedish TTS is not as developed as the English version. The Swedish voice sounds very robotic comparing to the bots available in English, and the amount of data that exists to train the bots is much more limited. The language barrier is causing trouble with the STT too, in Figure 11 if the bot used English instead of Swedish, all three tests would have most likely performed much higher in English. Another critical analysis that could have been done with more time would be to test how the bot handles different accents and nonnative speakers. This, however, would require more and bigger variate groups of people.

Since it is just a prototype improvements can always be done. Modifying the language would improve sounding and speech, which will result in the bot sound less robotic. By adding more examples in the intents that are used, it would be able to pick up more phrases from the user and lower the chance of ending up in the ANYTHING intent that picks up anything that bot is not looking for. By adding even more ways to say each question, there could technically be an infinite amount of ways a conversation can be held to improve the change of getting trustworthy answers. Dialogflow also has a built-in function where it says, "sorry I could not catch that can you repeat?" whenever it gets interrupted while talking and sometimes it gets stuck forget where in the narrative it came from. This could be fixed with more time and work, by looking into the connection between Narratory and Dialogflow and investigate the error is located.

All the tests in Section 4.2.2 were made by calling one or two persons at a time. There is still uncertainty if the bot work can perform to the same degree if there were a large number of calls made at the same time. By overloading the system, it could cause the VoIP to get more packet loss and worsen the quality of the call. In the current VoIP integration, there is implemented a maximum time limit for how long an answer can be. If the user overextends that limit, the agent will continue on. To make the phone call sound more realistic, this is something that should be changed into a dynamic limit to wait for it to be quiet for a specific time instead of having a static time limit. Since the bot is made for medical use the fact that it just moves on to the next question if not receiving an answer in time could be a flag that something serious is wrong and the user can not talk on their own, but right now the agent(Y) does not set any flag for a blank answer. A severe warning flag should be activated, especially if multiple or all questions are left without a solution.

5.2 Comparison To Related Work

As previous work mentioned in section 3.5, the kitchen-assistant, test-users experienced issues with the natural sounding speech, miss pronunciation, and slow response time from the system. The kitchen-assistant provides more functionality and more features than call-up assistant does. Though more

functionality might be a drawback for the system and why the test-users experience slow response time. As mentioned before, the major drawback for the call-up assistant is the language barrier since the Swedish language is not improved enough. The kitchen-assistant uses pyttsx3 module from Python library, which can have a big impact on why the speech sounds more 'robot-like', and since it is built on English there is a chance that Google's TTS would provide better quality that sounds more 'human-like'.

5.3 Social Requirements

For this project, it is essential to review ethics. The number of ethical laws is growing more than ever and is a big concern for big, fast-growing companies. All is about integrity, and the importance of not lacking private information about company users, violating the law can lead to big consequences. It can make a significant impact on the company. For instance, the new European law GDPR shows that validation, security, and personal data is of great importance, especially in this project, since it aims towards the medical field. Since every call gets logged and the conversation is stored, it is important to have implemented a database with secure database-rules, especially when it comes to the medical field. Some calls may contain more personal data than previous calls, so it is crucial to make sure that only authenticated users can access it. Another vital part to bring is the implementation of the dialog. From the end-user perspective, it can be uncomfortable to speak with an AI. Therefore it is essential as a company to provide information regarding the call. Such as introducing the concept of the product and the purpose of the call. If the end-user gets to know why the calls are made, more significant interest will come through, which could lead to a more reliable and valid conversation. It is essential to provide information about what data will be stored for further analysis. Therefore it is a lower chance that the end-user tells private information that should not be said.

6 Conclusion

In the beginning, this work was divided individual components that an autonomous call up uses. Then the work got split into two different thesis were this one would focus on how to make a realistic dialog and company provider that would suit this case the most. One of the first thing that was decided was to use an agile workflow. There were no precise requirements for the product from the company that came up with the idea. The agile workflow was the right decision because the process could continue and even gave the report more broad research when a discovery was made during the research state.

In the end, the decision on which TTS and STT provider to use fell on Google. This ended up being more useful than first expected because of the company being able to provide a Narratory an API built on Google Dialogflow that made it more plausible to make realistic dialog. Dialogflow, in itself, made use of Google's a large amount of data to train an agent to recognize phrases using NLP. In the parallel thesis that focused on integrating the web application, VoIP, and database, the decisions integrated well with Google. The database choice landed on Firebase, which is also made by Google, and the VoIP service landed on Voximplant, which has an integration with Dialogflow called Dialogflow Connector. Google happened to become somewhat of an umbrella for most of the services simplifying integration between all components when creating the prototype because some parts were already pre-integrated.

According to the results compared with the goals, conclusions are drawn that with more in-depth research, analysis of each technique, and applied methods such as WER and MOS, the most suitable cloud provider for this work could be found. This choice of cloud provider made it possible to keep up the work and start aiming for the prototype. By bringing more building blocks to the system as Narratory and Dialogflow, the dialog could be created. Interweaving the dialog with the database and VOIP service from the other thesis made it possible to develop the prototype for the autonomous call.

As for future work, there are many potentials to make the bot from a COVID-19 specific agent to be able to do a lot more in the medical field. Even without making the chatbot broader to handle more tasks, there are still a lot of improvements that can be made with just the current bot. The product needs to be put and tested by people who need medical support regularly to improve the intents to see how they answer the questions. When the product has been out in the market for a while, there is the possibility to analyze the gathered data to see if people who have had trouble using a particular answer could be used to predict abnormalities with future patients.

Also, the dialog can be tested related to the user experience. It can be made by selecting a group of people and let them experience the dialog-environment by talking to the agent. After the conversation, each person will rate the dialog by answering some questions that later could be used to improve specific functionalities.

References

[1] Wärmegård E. Intelligent chatbot assistant: A study of integration with voip and artificial intelligence. Halmstad University, 2020.

[2] Thomas Davenport & Ravi Kalakota. The potential of artificial intelligence in healtcare. 6,2:94–98, 2019.

[3] Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17, 2015.

[4] Yaniv Leviathan & Yossi Matias, Google Duplex: An AI System for Accomplishing Real-World Tasks Over the Phone. 2018 May 8. [cited 2020 Feb 9] Google AI Blog [Blog on the Internet]. Available from: `https://ai.googleblog.com`.

[5] Amazon. Alexa User Guide: Learn What Alexa Can Do. [Cited 2020 Apr 15]. Available from: `https://www.amazon.com`.

[6] Amazon. Amazon Echo. [Cited 2020 Apr 15]. Available from: `https://www.amazon.com`.

[7] Peter Mell, Tim Grance, et al. The NIST definition of cloud computing. 2011.

[8] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, D Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28(13):2009, 2009.

[9] Keiko Hashizume, David G Rosado, Eduardo Fernández-Medina, and Eduardo B Fernandez. An analysis of security issues for cloud computing. *Journal of internet services and applications*, 4(1):5, 2013.

[10] Subashini Subashini and Veeraruna Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11, 2011.

[11] Habeger SD. Cloud Services[Photography] 2010 [Cited 2020 Mar 2]. Avaliable from: `https://commons.wikimedia.org/w/index.php?title=File:Cloud_Services.gif&oldid=400548358`. (CC BY 3.0) `http://creativecommons.org/licenses/by-sa/3.0/`.

[12] Sushil Bhardwaj, Leena Jain, and Sandeep Jain. Cloud computing: A study of infrastructure as a service (iaas). *International Journal of engineering and information Technology*, 2(1):60–63, 2010.

[13] Joel Gibson, Robin Rondeau, Darren Eveleigh, and Qing Tan. Benefits and challenges of three cloud computing service models. In *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, pages 198–205. IEEE, 2012.

[14] Anthony T Velte, Toby J Velte, Robert C Elsenpeter, and Robert C Elsenpeter. *Cloud computing: a practical approach*. McGraw-Hill New York, 2010.

[15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[17] Aditya Amberkar, Parikshit Awasarmol, Gaurav Deshmukh, and Piyush Dave. Speech recognition using recurrent neural networks. pages 1–4, 03 2018.

[18] Nabi J. Recurrent Neural Networks (RNNs). 2019 Jul 11 [Cited 2020 Mar 2]. In towardsdatascience [Blog on the Internet] Available form: `https://towardsdatascience.com`.

[19] Lopez D. Your Guide to Natural Language Processing (NLP). 2019 Jan 15 [Cited 2020 Feb 18]. In towardsdatascience [Blog on the Internet] Available from: `https://towardsdatascience.com`.

[20] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

[21] Fahim Muhammad Hasan, Naushad UzZaman, and Mumit Khan. Comparison of different pos tagging techniques (n-gram, hmm and brill's tagger) for bangla. In *Advances and innovations in systems, computing sciences and software engineering*, pages 121–126. Springer, 2007.

[22] Kalaiarasi Sonai Muthu Anbananthen, Jaya Kumar Krishnan, Mohd Shohel Sayeed, and Praviny Muniapan. Comparison of stochastic and rule-based pos tagging on malay online text. *American Journal of Applied Sciences*, 14(9):843–851, 2017.

[23] S R Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 10 1998.

[24] D sasirekha & E Chandra. Hidden Markov Model. Stanford; 2019 [Cited 2020 Feb 21]. Available from: `https://web.stanford.edu/~jurafsky/slp3/A.pdf`.

[25] Thorsten Brants. Natural language processing in information retrieval. In *CLIN*, 2003.

[26] D'Souza J. Learning POS Tagging & Chunking in NLP. 2018 [Cited 2020 Mar 2]. In GrayAtom [Blog on the Internet]. Available from: `https://medium.com/greyatom`.

[27] Lluís Màrquez, Xavier Carreras, Kenneth C Litkowski, and Suzanne Stevenson. Semantic role labeling: an introduction to the special issue, 2008.

[28] Tumisho Mokgonyane, Tshephisho Sefara, Phuti Manamela, Madimetja Manamela, and Thipe Modipa. Development of a speech-enabled basic arithmetic m-learning application for foundation phase learners. pages 794–799, 09 2017.

[29] D Sasirekha and E Chandra. Text to speech: a simple tutorial. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(1):275–278, 2012.

[30] Thierry Dutoit. In *An introduction to text-to-speech synthesis*, pages 129–133, 1997.

[31] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 373–376. IEEE, 1996.

[32] Caroline Henton. Challenges and rewards in using parametric or concatenative speech synthesis. *International journal of speech technology*, 5(2):117–131, 2002.

[33] Google Cloud. WaveNet and other synthetic voices. [Cited 2020 Feb 19] Available from: `https://cloud.google.com`.

[34] Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

[35] Michelle Cutajar, Edward Gatt, Ivan Grech, Owen Casha, and Joseph Micallef. Comparative study of automatic speech recognition techniques. *IET Signal Processing*, 7(1):25–46, 2013.

[36] Rainer E Gruhn, Wolfgang Minker, and Satoshi Nakamura. *Statistical pronunciation modeling for non-native speech processing*. Springer Science & Business Media, 2011.

[37] Urmila Shrawankar and Vilas M Thakare. Techniques for feature extraction in speech recognition system: A comparative study. *arXiv preprint arXiv:1305.1145*, 2013.

[38] Shreya Narang and Ms Divya Gupta. Speech feature extraction techniques: a review. *International Journal of Computer Science and Mobile Computing*, 4(3):107–114, 2015.

[39] Google Cloud. Cloud Computing Services. [Cited 2020 Mar 5]. Available from: `https://cloud.google.com`.

[40] Stéphanie Challita, Faiez Zalila, Christophe Gourdin, and Philippe Merle. A precise model for google cloud platform. In *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pages 177–183. IEEE, 2018.

[41] Google Cloud Platform. Cloud Text-To-Speech. [Cited 2020 Mar 5]. Available from: `https://cloud.google.com`.

[42] Google Cloud Platform. Cloud Speech-To-Text. [Cited 2020 Mar 12]. Available from: `https://cloud.google.com`.

[43] Sajee Mathew and J Varia. Overview of amazon web services. *Amazon Whitepapers*, 2014.

[44] Amazon Web Services. What Is Amazon Polly? [Cited 2020 Mar 11]. Available from: `https://aws.amazon.com`.

[45] Amazon Web Services. Amazon Polly Pricing. [Cited 2020 Mar 14]. Available from: `https://aws.amazon.com`.

[46] Amazon Web Services. What Is Amazon Transcribe? [Cited 2020 Mar 11]. Available from: `https://aws.amazon.com`.

[47] Amazon Web Services. Amazon Transcribe Pricing. [Cited 2020 Mar 20]. Available from: `https://aws.amazon.com`.

[48] Ahmed Ali and Steve Renals. Word error rate estimation for speech recognition: e-WER. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 20–24, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[49] Kenny M. Rev Automated Transcription: Benchmarking Process and WER Calculations. 2019 [Cited 2020 Mar 6]. In rev Blog [Blog on the internet]. Available from: `https://www.rev.com/blog/`.

[50] Robert C Streijl, Stefan Winkler, and David S Hands. Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227, 2016.

[51] F. Ribeiro, D. Florêncio, C. Zhang, and M. Seltzer. Crowdmos: An approach for crowdsourcing mean opinion score studies. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2416–2419, May 2011.

[52] Julia Cambre, Jim Maddock, Janice Tsai, Jessica Colnago, and Jofish Kaye. Choice of voices: A large-scale evaluation of text-to-speech voice quality for long-form content.

[53] Google. Agents. [Cited 2020 APR 15]. Available from: `https://cloud.google.com`.

[54] Twilio. What is a Webhook?. [Cited 2020 Apr 18]. Available from: `https://www.twilio.com/docs`.

[55] Google Cloud. Webhook service. [Cited 2020 Apr 19]. Available from: `https://cloud.google.com`.

[56] Ludvig Linse. Narratory. https://narratory.io/docs/building-blocks. [Online; accessed 8-april-2020].

[57] Ludvig Linse.Fallbacks and error handling. [Cited 2020 Apr 18]. Available from: https://narratory.io/.

[58] Ludvig Linse. Advanced turns - DynamicBotTurn and BridgeTurn. [Cited 2020 Apr 18]. Available from: https://narratory.io/.

[59] Rystedt B & Zdybek M. Conversational agent as kitchen assistant. KTH Royal Institute of Technology, 2018.

A
Appendix

A.1

Intent Detection Confidence						
	Ideal		Long answer		Background noise	
n	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
1	0,924	0,169941	0,274	0,269407	0,784	0,241413
2	0,96	0,089443	0,208	0,297187	0,794	0,22289
3	1	0	0,412	0,23435	0,726	0,198444
4	0,96	0,089443	0,262	0,362657	0,722	0,272984
5	0,76	0,43359	0,598	0,038987	0,78	0,216795

Webhook latency						
	Ideal		Long answer		Background noise	
n	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
1	361	116,8161	3135,4	2549,125	514,6	150,457
2	412,8	93,13807	4053	2115,321	526,8	58,40976
3	351,8	101,4061	1321,2	2058,395	417,4	70,71987
4	332,4	75,64589	3201,2	2463,62	417,4	77,60799
5	566	283,1157	1271,6	2080,806	1175,4	1599,982

Figure 12: Testing represented in raw data

A.2

Listing 1: Code example how a conversation works when the AI start

```
const init: DynamicBotTurn = {
  url: "URL to API"
}
const queryFeeling: BotTurn = {
  label: "FEELING",
  say: {
    text: ["Hur mår du idag?", "Hur mår du?"],
  },
  user: [
    {
      intent: nlu.feelingGood,
      bot: {
        say: ["Vad bra!", "Härligt att höra!"],
        goto: "ISOLATE"
      }
    },
```

```
    {
      intent: nlu.feelingBad,
      bot: {
        say: ["Vad tråkigt", "Tråkigt att höra", "Det va ju inte bra"],
        goto: "SYMPTOM"
      }
    },
    {
      intent: nlu.feelingOkey,
      bot: {
        say: ["Okej"],
        goto: "SYMPTOM"
      }
    },
    {
      intent: [...nlu.fever.examples, ...nlu.cold.examples, ...nlu.cough.examples],
      bot: {
        say: ["Vad tråkigt", "Tråkigt att höra", "Det va ju inte bra"],
        set: {
          symptom: true
        },
        goto: "CONTROL"
      }
    },
    {
      intent: ANYTHING,
      bot: [
        {
          cond: { retryCount: 0 },
          say: 'Jag hade lite svårt att förstå vad du sa.
          Enklast för mig är om du säger "Jag mår bra", "Jag mår sådär"
          eller "jag mår inte så bra"',
          repair: true
        },
        {
          cond: { retryCount: 1 },
          say: ["Okej", "uppfattat"],
          goto: "SYMPTOM"
        }
      ]
    }
  ]
}
```

A.3

Listing 2: Code example of intents

```
export const feelingGood: Intent = {
  examples: ["Jag mår bra", "Ganska bra", "bra",
  "helt okej", "det är bra", "mår fint", "fint", "jo det är bra"]
}
export const feelingBad: Intent = {
  examples: [
    "Jag mår dåligt",
    "Inte så bra",
    "Ganska dåligt",
    "Jag har en dålig känsla",
    "Dåligt",
    "Kasst",
    "Jag mår illa",
    "jag mår skit",
    "skit"
  ]
}
```

A.4

Listing 3: The agent

```
const agent: Agent = {
  agentName: "Name of agent",
  language: Language.Swedish,
  narrative,
  userInitiatives,
  bridges: ["Så", "Var var vi", "Jo"],
  narratoryKey: require("../narratory_credentials.json").narratoryKey,
  googleCredentials: require("../google_credentials.json"),
  logWebhook: "URL for webhook",
  logLevel : "ALL"
}
```

A.5

Listing 4: Process of fetching end-user name

```
import { db } from "../../util/firebaseServer"
const getDataFromDb = async <T>({
    collection,
    id,
}: {
    collection: string,
    id: string
```

```
}): Promise<T> => {
    try {
        const dbRef = db.collection(collection).doc(id)
        const snap = await dbRef.get()
        const data = snap.data()
        return data as T
    } catch (error) {
        console.log(error)
        return null
    }
}
export default (async (req, res) => {
    const sessionId = req.body.sessionId
    const sessionData = await
            getDataFromDb<Session>({ collection: "session", id: sessionId })
    const clientId = sessionData.clientId
    const clientData = await
        getDataFromDb<Client>({ collection: "client", id: sessionData.clientId })
    const clientName = clientData.name
    res.send({
        set: {
            firstName: clientName
        }
    })
})
```

Johannes Hägglund

Linus Lerjebo

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se