Puheentunnistuksen mahdollisuudet sovellusten hallinnassa

Google-rajapinnan hyödyntäminen puheen tunnistamisessa

LAHDEN AMMATTIKORKEAKOULU
Insinööri (AMK)
Tieto- ja viestintätekniikka
Kevät 2019
Henri Happonen

Tiivistelmä

Tekijä	Julkaisun laji	Valmistumisaika
Happonen, Henri	Opinnäytetyö, AMK	Kevät 2019
	Sivumäärä 35	

Työn nimi

Puheentunnistuksen mahdollisuudet sovellusten hallinnassa
Google-rajapinnan hyödyntäminen puheen tunnistamisessa

Tutkinto

Tieto- ja viestintätekniikan koulutus, ohjelmistotekniikka, Insinööri (AMK)

Tiivistelmä

Opinnäytetyön tavoitteena oli testata puheentunnistuksen ja syntetisoinnin mahdollisuuksia työn toimeksiantajalle Avenla Oy:lle. Avenla Oy tuottaa asiakkailleen räätälöityjä digitaalisen liiketoiminnan ja markkinoinnin palveluita, kuten ekstranet- ja intranet-toimintoja.

Puheentunnistuksella on suhteellisen pitkä historia, jo 60-luvulta lähtien, mutta viimeisen 10 vuoden aikana sen hyödyntäminen ja kehitys on lisääntynyt huomattavasti. Puheentunnistusta voidaan käyttää apuna esimerkiksi yksinkertaisten toimintojen suorittamisessa, joita ovat laitteille luotavat hälytykset ja muistutusmerkinnät. Tämä mahdollistaa tietoteknisten laitteiden käytön ihmisille, joille niiden käyttö on haastavaa. Puheentunnistuksella voidaan myös tehostaa erilaisten uusien älylaitteiden käyttöä.

Puheentunnistuksen ja syntetisoinnin testaamiseen käytettiin Googlen ilmaista puheentunnistusrajapintaa sekä Facebookin Wit.ai:n luonnollisen kielen prosessointialustaa (NLP). Googlen rajapinta toimii Javascript-koodin avulla Googlen Chrome-selaimessa. Wit.ai-rajapintaa voi käyttää useista eri ohjelmistoalustoista, kuten Javascriptin suoritukseen käytettävä Node JS. Rajapintaa voi myös käyttää verkkoselaiten ja serverien väliseen tiedonsiirtoon käytettyjen HTTP-pyyntöjen avulla. Wit.ai tunnistaa sille lähetetyistä viesteistä niiden merkityksen ja palauttaa sen kutsuvalle sovellukselle.

Opinnäytetyön aikana kehitetyn sovelluksen tarkoitus oli testata puheentunnistuksen ja syntetisoinnin mahdollisuuksia jatkokehitystä varten. Luodun sovelluksen puheentunnistuksen testaaminen onnistui suhteellisen hyvin, mutta kehityksen aikana selvisi, ettei Googlen ilmainen rajapinta toimi työssä parhaalla mahdollisella tavalla, sillä se asettaa rajoituksia sovelluksen käytettävyyteen, koska rajapinnalla on täysi tuki ainoastaan Googlen Chrome selaimessa.

Asiasanat

Puheentunnistus, puheensyntetisointi, wit.ai, NLP

Abstract

Author(s)	Type of publication	Published
Happonen, Henri	Bachelor's thesis	Spring 2019
	Number of pages	
	35	

Title of publication

Speech recognition possibilities for controlling software
Utilizing Google interface for speech recognition

Name of Degree

Information and Communications Technology, Software Engineering

Abstract

The aim of the thesis was to test possibilities of speech recognition and synthesis for Avenla Oy, develops digital business and marketing products products for their customers.

Speech recognition has a long history starting from the 1960s, but during the past 10 years its use and development has increased greatly. Speech recognition can for example make simple tasks like setting alarms or making calendar entries on devices easier. It can also enable the use of devices for people who have limitations in using devices normally. Speech recognition also simplifies the way we interact with new smart devices.

Speech recognition and synthesis testing was carried out with Google's free speech recognition interface and the Wit.ai natural language processing (NLP) platform owned by Facebook. Google's speech recognition interface is included in the Google Chrome web browser and it is used with Javascript code. The Wit.ai interface can be used from different software platforms like Node js. It can also be called with HTTP requests used between browser and server for data transfer from different applications. Wit.ai recognizes the meaning of a message it receives and returns it to the calling application.

The speech recognition software developed for the thesis was aimed to test possibilities of speech recognition for further development. In the future the software will be connected with a background service, where the developed application receives notifications. The user of the software should then be able to listen and acknowledge received notifications using voice recognition and synthesis.

Speech recognition testing succeeded as expected. However, during development it however came clear that the Google free interface has limitations about the ways it can be used, since it is only properly supported in the Google Chrome browser.

Keywords

speech recognition, speech synthesis, wit.ai, NLP

SISÄLLYS

1 JOHDANTO

Puheentunnistus ei ole uusi käsite, vaan ensimmäiset alkeelliset puheentunnistuksen sovellukset on toteutettu jo 50-luvulla. Puheentunnistus on kuitenkin kehittyneiden algoritmien ja älypuhelinten tuoman helppouden ja saatavuuden ansiosta alkanut yleistyä voimakkaasti viime vuosien aikana. (Boyd 2018.) Puheentunnistus ja puheen muuntaminen tekstiksi, eli puheen syntetisointi on monimutkainen prosessi ja nopeasti lisääntyvä ja kasvava tekniikan ala. Puheentunnistuksessa mikrofoni havaitsee puheen aiheuttaman analogisen signaalin ja tämän jälkeen analogia-digitaalimuunnin muuntaa äänen tietokoneen ymmärtämään digitaaliseen muotoon. (Grabianowski 2006b.)

Nykyinen puheentunnistus hyödyntää tekoälyä tunnistamaan puhetta lähes reaaliaikaisesti. Puheentunnistuksen käyttö tehostaa yritystoimintaa ja nopeuttaa esimerkiksi raporttien tekemistä ja käsittelyä. (Reference 2019.) Puheentunnistuksella voidaan myös helpottaa sellaisten ihmisten toimintaa, joille puhuminen on normaalisti hankalaa. Puheentunnistuksen avulla voidaan tunnistaa hankalasti ymmärrettävää puhetta ja muuttaa se muille selkeämmäksi ymmärtää. (Able Here 2018.) Puheentunnistuksen kehityksessä ja yleistymisessä suuressa roolissa on helppo saatavuus kehittäjille. Nykyisin useat eri yritykset, kuten Google ja Microsoft, tarjoavat omia puheentunnistusrajapintoja kehittäjien käyttöön. Tämä mahdollistaa kehittäjiä luomaan monimutkaisia sovelluksia ilman suuria panostuksia puheentunnistuksen perusteisiin. (Google Cloud 2019.)

Työn toimeksiantaja Avenla Oy on kiinnostunut puheentunnistuksen tuomista mahdollisuuksista. Avenla Oy tarjoaa digitaalisia liiketoiminnan ja markkinoinnin palveluita räätälöidysti asiakkailleen. Palveluihin kuuluu asiakkaiden tarpeiden mukaisesti toteutetut Internet-, extranet- ja intranet-ratkaisut sekä esimerkiksi ratkaisuja sovelluskehitykseen ja tietoturvaan. Tässä opinnäytetyössä on tarkoituksena toteuttaa puheentunnistukseen liittyvä tutkimus ja toteuttaa teoriaan pohjaten sovellus, joka osaa tunnistaa ja syntetisoida puhetta.

Tässä opinnäytetyössä on tarkoituksena toteuttaa puheentunnistukseen liittyvä sovellus, jonka avulla käyttäjä voi puheen ja tekstin avulla kommunikoida tekoälyn kanssa. Sovellus on prove of concept, jonka avulla on tarkoitus testata puheentunnistusta sekä syntetisointia erilaisten taustajärjestelmien kanssa sovelluksen jatkokehitystä varten.

2 PUHEENTUNNISTUS TUKENA TIETOTEKNIIKASSA

2.1 Puheentunnistuksen kehittyminen apuvälineeksi

Puheentunnistus ei ole tänä päivänä enää mitenkään uusi ilmiö (Pinola 2011). Bell Laboratioes kehitti 1950-luvulla Audrey-järjestelmän, joka tunnisti puhuttuja numeroita. Audrey-järjestelmä oli erittäin rajallinen, koska sen lisäksi, että se oli rajoittunut vain numeroiden tunnistukseen, se tunnisti vain yhden äänen. Audrey-järjestelmä oli kuitenkin ensimmäinen käytännön sovellus, joka kykeni tunnistamaan ihmisen. (Sonix Inc 2019.) Seuraava selkeä kehitysaskel nähtiin Pinolan (2011) mukaan IBM:n toimesta 1962, kun se esitteli Shoebox-koneen, joka tunnisti 16 englanninkielistä puhuttua sanaa.

Puheentunnistus kehittyi aluksi hyvin hitaasti ja pienissä askelissa. Aluksi tunnistus toimi vain tietyillä ennalta määrätyillä sanoilla, kunnes 1960-luvulla kehitettiin Markovin piilomalli, joka tuli suureen osaan puheentunnistusta 1980-lulla. (Boyd 2018.) Markovin piilomalli on tilastollinen malli, jossa havainnoitava data mallinnetaan sarjana lopputulemia. Mallissa ei tiedetä seurattavan ilmiön sisäisiä tiloja eikä siirtymiä niiden välillä. Markovin piilomallia on käytetty paljon esimerkiksi kryptografiassa, puheentunnistuksessa ja puheensyntetisoinnissa. Ennen Markovin piilomallia puheentunnistuksessa tunnistettiin yksittäisiä sanoja etsimällä puheesta tiettyjä äänen kuvioita. Markovin piilomallin avulla alettiin tutkia todennäköisyyttä siitä, että tuntematon ääni olisi sana. Markovin piilomallia käyttävät sovellukset alkoivat ennakoida edellisten äänteiden ja äänen kuvioiden perusteella seuraavien äänteiden todennäköisyyksiä. (Boyd 2018; Pinola 2011.)

Tekniikan kehitys on monessa asiassa muokannut yhteiskunnan yleisempiä toimintatapoja (Niinimäki 2015). Tekniikan kehittyessä ja lisääntyessä ei ole aina ihanteellista tai edes mahdollista käyttää perinteistä näppäimistöä ja hiirtä sovellusten ja laitteiden ohjaamiseen. Jatkuvasti kehittyvä puheentunnistus ja syntetisointi sekä tekoälyn kehitys nousevat jatkuvasti tärkeämpään rooliin uusien käyttöliittymien kehityksessä. (Lillback 2017.) Puheentunnistaminen tarkoittaa sitä, että tietokone osaa kuunnella ja tulkita sille puhuttua tai lähetettyä ääntä ja reagoida siihen. Puheen syntetisointi taas tarkoittaa sitä, että tietokone osaa puhua ääneen ihmiselle ymmärrettävällä tavalla. Puheen syntetisointi helpottaa esimerkiksi automatisoitujen viestien ja ilmoitusten ylläpitoa, koska ennalta tallennettujen äänitteiden sijaan kone voi synteettisesti toistaa halutun viestin, jota voidaan tarvittaessa muuttaa nopeallakin aikataululla. (Cryer 2018.)

2.1.1 Puheentunnistuksen tarjoamat hyödyt

Puheentunnistusta sekä puheen syntetisointia voi käyttää moniin eri tarkoituksiin. Sen avulla voi helpottaa ihmisten arkipäiväisiä askareita, helpottaa usean asian samanaikaista tekemistä ja mahdollistaa laitteiden käyttö ihmisille, joille tavallisten käyttöliittymien hallinta on vaikeaa tai mahdotonta. Puheentunnistus on hyvä vaihtoehto tilanteisiin, joissa erilaisia toimintoja on valittavana useita. Oikean toiminnan valitseminen laajasta valikoimasta graafisesti on vaikeaa, mutta puhumalla se onnistuu helposti ja nopeasti. (Järvinen 2017.) Puheentunnistusta voidaan käyttää laitteiden ohjaamisen lisäksi myös puheen tulkitsemiseen. Puheen tunnistukseen ja syntetisointiin liittyy oleellisesti käsitteet ”puheesta tekstiksi” ja ”tekstistä puheeksi”. Nämä ominaisuudet tarkoittavat puheen muuntamista tekstimuotoon ja tekstin muuntamisen puheeksi. (Beal 2019.)

2.1.2 Digiavustajat

Monille puheentunnistus on tullut osaksi arkipäivää älypuhelinten kautta, sillä kaikissa moderneissa älypuhelimissa on mukana digiavustaja, joka pystyy auttamaan käyttäjää puhelimen käytössä ja suorittamaan rutiinitoimenpiteitä ilman, että käyttäjän täytyy koskea laitteeseen. Siri, Cortana ja Google Now ovat yleisimmät ja tunnetuimmat digiavustajat. Niistä kaikki osaavat tehdä perustoimenpiteet, ja ne nopeuttavat yksinkertaisten asioiden tekemistä. Digiavustajat osaavat esimerkiksi lisätä kalenterimerkintöjä, lukea karttaa tai lähettää käyttäjän sanelemia viestejä. (Sulopuisto 2016.)

Kaikki nämä ominaisuudet ovat hyödyllisiä esimerkiksi autoa ajaessa, jolloin auton kuljettajan huomio on liikenteen seuraamisessa. Tällöin digiavustajaa voi käskyttää nopeilla ja yksinkertaisilla komennoilla, eikä kuljettajan tarvitse irrottaa huomiota liikenteestä ja auton ohjaamisesta. (Barr 2019.) Lisäksi puheentunnistusta ja digiavustajia voidaan käyttää kiireisten ihmisten auttamiseen. Esimerkiksi lääkärit voivat puheentunnistuksen avulla keskittyä potilaiden kanssa kommunikointiin sen sijaan, että lääkärin kaikki aika menisi muistiinpanojen kirjoittamiseen tapaamisen aikana ja sen jälkeen. Puheentunnistus ja puheen syntetisointi voivat muutenkin auttaa lääkärin työtä. Mikäli potilaan ja hoitohenkilökunnan välillä ei ole yhteistä kieltä, digitaalinen avusta voi toimia reaaliaikaisena tulkkina lääkärin työn helpottamiseksi. (Mäntylä 2018.)

2.1.3 Erityisryhmät

Tavalliselle ihmiselle puheentunnistus ja syntetisointi ovat lähinnä mukavuustekijä, jolla pyritään helpottamaan ja nopeuttamaan asioita. Monille ihmisille mukavuus on kuitenkin sivuseikka. Esimerkiksi näkövammaisille sekä ihmisille, joiden käsien käyttö on rajoittunutta, mukavuuden sijaan on kyse ylipäänsä kyvystä käyttää laitteita arkipäiväisiinkään asioihin. (Rajala 2019.) Suomalainen YLE pyrkii kehittämään omia palveluitaan sopivammaksi kuulo- ja näkövammaisille ihmisille. Yle on esimerkiksi laskenut taustamusiikin voimakkuutta ohjelmissaan, koska se häiritsee ihmisiä, joiden on muutenkin vaikea kuulla puhetta. Lisäksi YLE pyrkii parantamaan tv-lähetysten livetekstityksiä, jotta myös kuurot ihmiset pääsevät seuraamaan helpommin, mitä suorissa lähetyksissä tapahtuu. (YLE 2019.)

2.2 Puheentunnistuksen haasteet

Puheentunnistus kehittyy voimakkaasti, mutta siinä on edelleen paljon parantamisen varaa eri kielien ja puhujien kohdalla. Yksi puheentunnistuksen hankaluuksista on se, että jokainen meistä puhuu hieman eri tavalla. Jokaisella on oma ääni ja tyypillinen puhenopeus, ja tämän lisäksi on olemassa paljon eri kieliä. (Eugene 2019.) Yksi haaste puheentunnistuksen kehityksessä on se, että on olemassa valtava määrä eri variaatioita, joilla saman asian voi sanoa. Tämä ongelma pätee kaikkeen tekoälyn kehitykseen. Ihmisen on helppo reagoida pieniin muutoksiin äänen painoissa tai jopa ymmärtää vierasta murretta, mutta tietokoneelle nämä erot ovat kuitenkin erittäin laajoja. (Sivasankaran 2017.)

Lisäksi toimintoja hankaloittaa kielten määrä. Eri kielet ovat rakenteellisesti erilaisia. Englannin kieli on valta-asemassa, koska se on yksi puhutuimpia kieliä ympäri maailmaa ja se on laajasti käytössä kansainvälisenä kielenä esimerkiksi kaupankäynnissä. (Lingua english 2015.) Pienten kielten, kuten suomen kielen kohdalla kehitys on paljon hitaampaa. Englannin kielellä suoria lähetyksiä on voitu tekstittää automaattisesti jo pitkän aikaa, mutta suomen kielen kannalta tilanne on vasta viime vuosina alkanut parantua Suomessa. (Rajala 2019.)

2.3 Erilaiset puheentunnistuksen muodot

Puheentunnistusta ja syntetisointia voidaan tehdä usealla eri tavalla. Puheentunnistusohjelma voi tunnistaa sille kehittäjän itse määräämiä yksittäisiä käskyjä tai se voi toimia

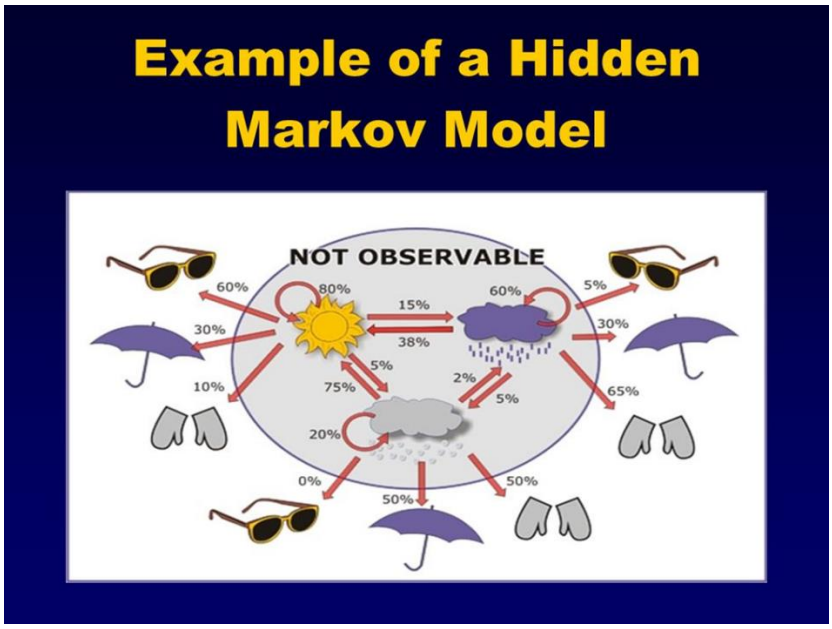
pilvipalvelussa ja tunnistaa tuhansia eri sanoja eri kielillä. Puheentunnistus- ja syntetisointisovelluksia kehittäessä on tärkeää miettiä tarkkaan, mikä on sovelluksen käyttötarkoitus. (Mozer 2015.) Esimerkiksi puhelinpalveluiden yhteydessä toimiva sovellus, joka luettelee käyttäjälle eri toimintoja ja niihin liittyviä numeronäppäimiä, ei välttämättä tarvitse tukea tuhansien sanojen tunnistukseen, vaan on tehokkaampaa keskittyä yksittäisten sanojen tunnistamiseen tehokkaasti. Näin saadaan nostettua sovelluksen toimintavarmuutta ja vähennettyä turhaa kuormaa sovelluksessa ja sovellus toimii tehokkaasti monilla eri käyttäjillä puhetyylistä riippumatta. (Call Centre Helper 2017.) Joidenkin sovellusten kannalta on tärkeämpää tunnistaa tuhansittain sanoja. Tällöin on parempi keskittyä tehokkaan puheen tunnistus algoritmin käyttöön, joka oppii tunnistamaan käyttäjän puheen. Tämän kaltaiset sovellukset voivat olla tehokkaita tuttujen käyttäjien kohdalla, mutta niiden tehokkuus laskee nopeasti, jos sovellus ei tunnista puhujan ääntä. (Grabianowski 2006.)

Puheentunnistus voi olla myös joko jatkuvaa tai eroteltua. Eroteltu puheentunnistus perustuu siihen, että jokaisen sanan välissä on selkeä ero. Tämä helpottaa puheentunnistussovelluksen toimintaa huomattavasti, mutta ei ole luonnollista ihmisille. Usein ihmiset puhuvat mieluummin itselleen luonnollisella tavalla, ilman taukoja ja tästä johtuen modernit puheentunnistussovellukset kehitetään jatkuvan puheentunnistukseen sopiviksi. (Grabianowski 2006.)

2.3.1 Markovin piilomalli

Markovin piilomallilla voidaan havainnoida esimerkiksi ulkona olevaa säätä sen mukaan, minkälaisia vaatteita ulkona liikkuvilla ihmisillä on päällään. Mallilla voidaan esimerkiksi todeta, että ulkona sataa vettä, kun havaitaan, että ulkona liikkuvat ihmiset ovat pukeutuneet lämpimästi vettä pitäviin vaatteisiin ja suurella osalla on sateenvarjo. (Kaplan & Biggin 2012.)

Puheentunnistuksessa tunnistettava puhe jaetaan foneemeihin, eli puheen pienimpiin tunnistettaviin osiin, joista sitten Markovin piilomallin avulla etsitään todennäköisesti kontekstiin sopivimmat vaihtoehdot. Näin puheentunnistus sovellus saa Markovin piilomallin avulla päätellä puhenäytteen sisällön. (Tieteen termipankki 2019.)



Kuva 1. Esimerkki Markovin piiilomallista (Rodgers 2016)

Kuvassa 1 on esimerkki Markovin piilomallista. Sen keskellä olevan ympyrän sisällä on seurattavaan ilmiöön liittyviä muuttujia, joiden todellista tilaa ei tiedetä eikä niihin liittyviä tilanmuutoksia. Ympyrän ulkopuolella on havaittavia lopputulemia, joiden perusteella voidaan päätellä, millä todennäköisyydellä havaittava ilmiö on tietyssä tilassa. Puheentunnistuksessa kuvan mukaiset havaittavat ilmiöt olisivat puheen akustisia piirteitä. Näiden akustisten piirteiden avulla päätellään foneemit, joita ne todennäköisimmin vastaavat (Le 2016).

2.3.2 Puheentunnistus paikallisesti

Puhetta voidaan tunnistaa ja syntetisoida paikallisesti käyttäjän laitteella. Tällöin kehittäjä määritelee puheentunnistussovelluksen tunnistamat sanat paikalliseen tietokantaan tai sovelluksen lähdekoodiin. Tämä voi olla tehokas tapa, mikäli halutaan ohjata rajallista määrää erilaisia komentoja tai antaa esimerkiksi ääneen puhuttuja ohjeita käyttäjälle (Mozer 2011). Paikallisesti toteutettu puheentunnistus vaatii kehittäjältä runsaasti työtä erityisesti sovelluksen kasvaessa. Tämä kuitenkin mahdollistaa nopean vasteajan, koska kaikki tunnistus ja syntetisointi tapahtuu laitteella. Lisäksi paikallisesti toteutettu sovellus on tietoturvan kannalta parempi ratkaisu, kun sovellus käsittelee kaiken datan paikallisesti. Näin sovelluskehittäjällä on täysi päätäntävalta siitä, miten sovellus käsittelee dataa. Tällöin esimerkiksi sovellus, jota käytetään arkaluontoisissa tilanteissa, ei vuoda mitään sovelluksen käsittelemää dataa ulkopuolisille. (Mozer 2015.)

Vaikka paikallinen puheentunnistus ei ole aina paras ratkaisu, on sille silti useita hyviä syitä. Paikallisesti toteutettu tunnistus ja syntetisointi tekee sovelluksesta riippumattoman

internet-yhteyksistä ja tämän lisäksi sovelluksen tehokkuus paranee. Esimerkiksi älypuhelimet säästävät virtaa, kun sovelluksen ei tarvitse jatkuvasti lähettää tietoa internetiä hyödyntämällä, tai sovellus, jonka on tarkoitus hallita laitteen kameraa, ei välttämättä tarvitse yhteyttä pilveen, vaan sovellus voi toimia laitteella itsenäisesti. (Mozer 2011; Mozer 2015.)

2.3.3 Puheentunnistus pilvessä

Pilvessä toteuttava puheentunnistus ja syntetisointi on mahdollistanut monimutkaisten sovellusten kehittämisen, sillä pilvessä tapahtuva puheentunnistus mahdollistaa sen, että puheentunnistus voidaan helposti upottaa osaksi useita eri sovelluksia. Sovelluskehittäjien ei tarvitse toteuttaa itse kaikkea toimintaa, jolloin sovellusten kehittämisessä säästyy runsaasti aikaa. Käyttäjiä tämä hyödyttää siinä, että sovellukset monipuolistuvat ja niiden käyttö helpottuu. (Lawton 2018.)

Pilvipalveluiden hyödyntäminen puheentunnistuksessa ja syntetisoinnissa sisältää myös ongelmia, koska sovellukset ovat riippuvaisia verkkoyhteydestä ja kehittäjien täytyy olla tarkkoja siitä, että sovellukset eivät kuluta liikaa akkua (Mozer 2015). Lisäksi pilvessä tapahtuva puheentunnistus toteutetaan usein ulkoisten palvelun tarjoajien kautta, kuten Googlen tai Microsoftin avulla. Tällöin tunnistus tapahtuu ulkoisen tekijän toimesta ja tämän seurauksena kehittäjä ei ole enää ainoa tekijä, jolla on pääsy käyttäjän tietoihin. Monesti tämä tiedonkäsittely on täysin harmitonta ja sitä hyödynnetään palvelun kehittämiseen, mutta tiedon väärinkäyttö on kaikesta huolimatta helpompaa, kuin paikallisessa tunnistuksessa. (Heires 2017.)

Ulkoisten palveluntarjoajien käyttäminen tuo myös hyviä mahdollisuuksia, koska Google sekä Microsoft tarjoavat kehittäjille mahdollisuuden hyödyntää eri palveluitaan. Esimerkiksi Google voi helposti yhdistää oman Google-kääntäjä palvelunsa kehittäjien sovelluksiin tai antaa laajat käyttömahdollisuudet Android-käyttöjärjestelmään. (Google Cloud 2019.) Microsoft taas omistaa Windows-käyttöjärjestelmän, Microsoft Office-tuotteet sekä Azure-palvelun. Näitä palveluita hyödyntämällä kehittäjät saavat laajennettua sovelluksensa toimintaa ja yhteensopivuutta nopeasti ja helposti. (Microsoft Azeure 2019.)

3 GOOGLE RAJAPINNAN KÄYTTÖ PUHEENTUNNISTUKSESSA

3.1 Googlen puheentunnistuksen rajapinta

Google on yksi suurimpia ja tunnetuimpia palvelun tarjoajia puheen tunnistuksen kehityksessä. Googlen suuri etu on siinä, että se omistaa laajan valikoiman erilaisia sovelluksia monilla eri tietotekniikan osa-alueilla. Tämä laaja sovellustarjonta takaa sen, että Google voi hyödyntää tehokkaasti kehittämiään teknologioita eri sovellusten välillä. Tämä myös auttaa houkuttelemaan kehittäjiä Googlen sovellusten pariin. (Google Cloud 2019.)

Kaupallisten palvelujen lisäksi Google tarjoaa myös ilmaisia rajapintoja sovellusten kehittämiseen. Yksi näistä rajapinnoista on Googlen Web Speech API. Se on rajapinta, joka mahdollistaa kehittäjiä tuomaan puheentunnistus- ja syntetisointiominaisuuksia nettisivuille. Vaikka rajapinta onkin ilmainen, se tarjoaa kehittäjille melko monipoliset käyttömahdollisuudet. Rajapinta tukee sekä serverillä että asiakkaan päässä tapahtuvaa puheentunnistusta ja syntetisointia. Rajapinta tukee sekä yksittäisten sanojen tai käskyjen tunnistusta tai jatkuvaa puheentunnistusta. Tunnistettu puhe palautetaan takaisin kutsuvalle sivulle hypoteesilistana. (Shires & Wennborg 2012.)

Yksittäisten käskyjen tunnistamista voisi käyttää esimerkiksi ennalta määrättyjen komentojen suorittamiseen. Sovellukseen voitaisi esimerkiksi määritellä komento ”lähetä”, joka lähettäisi tekstikenttään kirjoitetun viestin pikaviestintäsovelluksessa. Jatkuva puheentunnistus taas mahdollistaa esimerkiksi pitkien viestien sanelun. Tämä voi olla hyödyllistä vaikkapa sähköpostien lähettämisessä, jos lähettäjä on kiireinen eikä pysty kirjoittamaan viestiä. (Shires & Wennborg 2012.)

3.2 Käyttökohteet

Googlen puheentunnistusrajapinta tukee monenlaisia eri käyttötarkoituksia. Puheen tunnistusta ja syntetisointia voidaan hyödyntää nettisivuilla navigoinnin helpottamiseksi tai esimerkiksi lukemaan ääneen osia nettisivun sisällöstä.

Rajapintaa käyttämällä voi esimerkiksi toteuttaa seuraavanlaista toimintaa nettisivuilla:

- nettihakujen tekeminen puhumalla
- puheella ohjattava käyttöliittymä
- puheella täytettävät täyttökentät
- puheen kääntäminen toiselle kielelle

- ääneen luetut ja puhutut ajo-ohjeet

(Shires & Wennborg 2012)

Esimerkiksi äänellä täytettävät syöttökentät olisi mahdollista saada toimimaan niin, että käyttäjä puhuu ääneen sivulle haluamansa asian, ja sovellus poimii puheesta tarvittavat tiedot ja täyttää hakukentät automaattisesti. Puheen kääntäminen toiselle kielelle mahdollistaisi kahden eri kieliä puhuvien ihmisten välisen keskustelun. Tämä mahdollistaa sen, että toinen keskustelun osapuoli voi puhua omalla äidinkielellään sovellukselle, joka sitten kääntää puheen tekstiksi ja tarvittaessa lukee tekstin ääneen. Äänellä toimivat ajo-ohjeet taas helpottavat ohjeiden seuraamista, kun kuski voi kuunnella ajo-ohjeita kartan katsomisen sijaan. Lisäksi puheella toimivat ajo-ohjeiden haut vähentävät kuskin tarvetta käyttää käsiään ohjeiden hakemiseen. Tällöin kuskin keskittyminen pysyy tiessä ja liikenneturvallisuus paranee. (Shires & Wennborg 2012.)

3.3 Vaatimukset ja rajoitukset

Jotta puheentunnistusrajapintaa voi käyttää, täytyy kehittäjän sekä käyttäjän ottaa huomioon tiettyjä rajapinta kohtaisia rajoituksia. Kehittäjän näkökulmasta on huomioitava se, että tätä rajapintaa kutsutaan Javascript-koodin kautta. Tästä johtuen rajapintaa voi käyttää erilaisten nettisivujen kanssa. (Shires & Wennborg 2012.)

Rajapinnan hyödyntäminen vaatii selaimelta tuen Webkit-Speech-Recognition komponentille. Tämän takia käyttäjän näkökulmasta on suositeltavaa käyttää Google Chrome-selainta, koska se on ainoa selain, joka tukee Webkit-Speech-Recognition komponenttia täysin. Firefox- sekä Edge-selaimet tukevat komponenttia osittain, sillä niistä löytyy tuki puheen syntetisoinnille, mutta niistä puuttuu tuki puheentunnistukselle. Kehittäjä voi ilmoittaa käyttäjälle epäyhteensopivasta selaimesta liittämällä sivulle kuvan 2 mukaisen testikoodin, joka testaa mitä osia Webkit-komponentista käyttäjän selain tukee. (Shires 2019.)

```
if ('speechSynthesis' in window) {
 // Synthesis support. Make your web apps talk!
}

if ('SpeechRecognition' in window) {
  // Speech recognition support. Talk to your apps!
}
```

KUVA 2. Testi koodi Webkit-Speech-Recognition komponentin tuelle (Bidelman 2019)

3.4 Yksityisyys ja turvallisuus

Googlen puheentunnistus rajapinnassa on määritelty yksityisyyttä ja turvallisuutta koskien tiettyjä ehtoja, joita kehittäjän on noudatettava. Rajapintaa käytettäessä käyttäjälle on selkeästi ilmoitettava ja käyttäjän on erikseen hyväksyttävä sivulla tapahtuva puheen tunnistus ennen kuin puheentunnistuksen voi aloittaa. Hyväksynnän antaminen voi kattaa esimerkiksi erillisen mikrofonipainikkeen painamisen tai vastaamisen erilliseen kysymykseen koskien puheen tunnistuksen aktivoimista. Lisäksi kehittäjän on selkeästi ilmoitettava sivun käyttäjälle, milloin puheen tunnistus on käytössä. Tämän voi toteuttaa esimerkiksi puhekuplalla tai jollain muulla visuaalisella tavalla. (Shires & Wennborg 2012.)

Kehittäjän on myös pidettävä huoli siitä, ettei rajapintaa käytetä väärin tallentamaan käyttäjän puhetta ilman käyttäjän hyväksyntää ja selkeää tietoisuutta puheentunnistuksesta. Tämä tarkoittaa kehittäjän näkökulmasta sitä, että sivun toteutuksessa on huomioitava käyttäjän toimet sivulla. Jos käyttäjä vaihtaa esimerkiksi selaimen välilehteä tai käyttää jotakin muuta ohjelmaa samaan aikaan, jolloin nettisivu ei ole aktiivisena, niin puheentunnistus täytyy pysäyttää. (Shires & Wennborg 2012.)

4 WIT.AI-RAJAPINTA

4.1 Wit.ai

Tekoälyn käyttö apuvälineenä lisääntyy lähes jokaisella eri toimialalla, ja se on yksi nopeimmin kehittyvistä tekniikan aloista. Tekoäly mahdollistaa tiedon nopeamman käsittelyn ja välittämisen sekä mahdollisuuden hakea tietoa luonnollisen keskustelun omaisesti nettisivuilta. Tämä näkyy esimerkiksi monien lehtien haluna tuoda tekoälyyn perustuvia keskusteluominaisuuksia omiin palveluihinsa ja sovelluksiin. Hyvän ja tehokkaan tekoälyn kehitys on kuitenkin raskasta ja harvalla kehittäjällä on resursseja tai osaamista toteuttaa kokonaista tekoälyalustaa itsenäisesti. (McClendon 2016.)

Wit.ai on Facebookin kehittämä ja omistama luonnollisen kielen prosessointiin (NLP, natural language processing) tarkoitettu tekoäly alusta, joka auttaa kehittäjiä ja palvelun tarjoajia pääsemään nopeammin liikkeelle NLP-sovellusten kehityksessä. Se helpottaa kehittäjiä piilottamalla monimutkaiset yksityiskohdat, joita tekoälyn luontiin ja käyttöön liittyy. Tämä mahdollistaa kehittäjiä keskittämään resurssit toimivien sovellusten kehittämiseen ja haluamiensa ydin ominaisuuksien toteuttamiseen. Sen avulla voi kehittää sovelluksia, joille voi puhua luonnollisesti ja joka osaa puhua takaisin käyttäjälle. Wit.ai kehittyy koko ajan käytön mukana ja näin ollen sen käyttö tehostuu jatkuvasti ja hyödyttää kaikkia kehittäjiä. (Wit.ai 2019a.)

Merkitys ja entiteetti ovat keskeisiä termejä Wit.ai:n käyttämässä NLP prosessissa. Merkitys tarkoittaa käyttäjän sanoman todellista merkitystä. Tämä tarkoittaa sitä, että sovellus tunnistaa käyttäjän kysymyksestä haluaako käyttäjä esimerkiksi ajo-ohjeita vai tiedusteleeko käyttäjä säätä. Wit.ai Entiteetti taas tarkoittaa käyttäjän Wit.ai:lle lähettämään viestiin liittyviä muuttujia. Tämä tarkoittaa esimerkiksi tietoa siitä, minkä paikan sään käyttäjä haluaa tietää tai minkä lämpötilan käyttäjä haluaa Wit.ai:n säätävän toimiston huoneenlämpötilaksi. Sovelluksen kehittäjän on huomioitava se, että koska Wit.ai on NPL palvelu, se ei kuitenkaan osaa vastata käyttäjän kysymykseen. Esimerkiksi käyttäjän kysyessä säätä, Wit.ai palauttaa sovellukselle tiedon siitä, että käyttäjä kysyi säätä ja minkä paikan säätä käyttäjä tiedusteli. Wit.ai ei kuitenkaan kerro esimerkiksi tietyn kaupungin lämpötilaa, vaan kehittäjän on itse toteutettava tämä toiminnallisuus sovelluksessa. Wit.ai:n vahvuus on kuitenkin se, että käyttäjä voi sanoa saman asian erilaisilla variaatioilla ja Wit.ai pystyy silti tunnistamaan käyttäjän todellisen sanoman. (Filipeguelber 2017.)

4.2 Wit.ai:n käyttökohteet

NLP:n kehityksessä yksi tärkeä osa-alue on luonnollisen keskustelun aikaansaaminen käyttäjän ja sovelluksen välillä. Tämä prosessi vaati paljon resursseja ja ymmärrystä ihmiselle luonnollisesta tavasta toimia ja kommunikoida. Tätä prosessia kutsutaan luonnollisen kielen prosessoinniksi. Wit.ai tarjoaa kehittäjille helpon ja tehokkaan alustan toteuttaa luonnollisen kielen prosessointia. Tämä on tehokas tapa parantaa sovellusten keskustelujen luonnollisuutta ja siksi se onkin kriittisessä osassa Wit.ai:n käytössä. Luonnollisen kielen prosessointi on tärkeä osa kaikkea tekoälyn kehitystä, jossa on tarkoitus mahdollistaa ihmisen luonnollinen kommunikointi sovelluksen kanssa. (Zohaib 2016.)

Wit.ai:ta voi käyttää myös tehostamaan ja helpottamaan kodin automaatiota. Kodeissa lisääntyvät älylaitteet mahdollistavat kodin hallinnan ja valvonnan tehokkaasti. Wit.ai helpottaa tätä prosessia mahdollistamalla sovelluskehittäjiä sekä harrastelijoita kehittämään omia helppokäyttöisiä ja tehokkaita hallinta mahdollisuuksia erilaisille kodin älylaitteille. Wit.ai mahdollistaa esimerkiksi älylaitteiden ohjaamisen äänellä tai itsekehitetyn helppokäyttöisen käyttöliittymän kautta. (Krishnan 2018.)

Wit.ai on mahdollista liittää useisiin eri tyyppisiin sovelluksiin. Siihen löytyy valmiina dokumentaatiota ja esimerkki projekteja Node js, Python sekä Ruby ympäristöihin. Tämä mahdollistaa aloittelijoille helpon liikkeelle pääsyn. Wit.ai ei kuitenkaan ole rajoittunut vain tiettyihin ympäristöihin, vaan sitä voi kutsua myös HTTP-pyyntöjen avulla mistä vain ympäristöstä. HTTP-pyyntöjen avulla Wit.ai on mahdollista upottaa osaksi nettisivuja tai mobiilisovelluksia. Lisäksi Wit.ai on mahdollista yhdistää Facebook Messenger sovellukseen, jolloin sen avulla on mahdollista luoda keskustelu botteja, jolle on nopeasti saatavissa suuri määrä käyttäjiä. (Wit.ai 2019b.)

4.3 Rajapinnan kuvaus

Wit.ai rajapintaa kutsuttaessa on kutsun yhteydessä välitettävä sovelluskohtainen merkki jono, jota Wit.ai käyttää käyttäjän ja sovelluksen tunnistamiseen. Tämä on turvatoimenpide, joka estää luvattoman pääsyn muiden kehittäjien sovelluksiin. Tunnistukseen käytettäviä sovellusavaimia on kahdenlaisia. Asiakkaan pään avain mahdollistaa esimerkiksi omalta nettisivulta suoritettavat pyynnöt, jotka mahdollistavat ainoastaan tiedon lukemisen. Rajapinnalle tehtävien kutsujen avulla on mahdollista myös suorittaa kirjoittavia toimenpiteitä sovelluksen muuttamiseksi. Sovelluksen muuttaminen rajapinta kutsujen avulla suositellaan tehtävän kehittäjän omalla palvelimella käyttämällä palvelimen pään avainta.

Selaimella tehtävä sovelluksen muokkaaminen palvelimen avainta käyttämällä mahdollistaa palvelin avaimen päätymisen väärin käsiin. (Wit.ai 2019c.)

Käyttäjän tunnistamisen lisäksi jokaisessa rajapinta kutsussa on määritettävä versiotieto. Versiotieto on päivämäärä, joka määrittää rajapinnan version. Versio tieto on tärkeä siksi, että Wit.ai:n kehittyessä, saattaa osa sen toiminoista muuttua niin, että vanhat versiot eivät ole enää yhteensopivia. Versiotiedon avulla sovellus voi jatkaa toimintaa, vaikka se käyttääkin vanhentunutta Wit.ai versiota. Wit.ai rajapinta vastaa saamiinsa onnistuneisiin kutsuihin JSON muodossa ja epäonnistuneisiin pyyntöihin rajapinta palauttaa tekstimuotoisen vastauksen. (Wit.ai 2019c.)

KUVA 3. Esimerkki Wit.ai:n palauttamasta JSON vastauksesta (Wit.ai 2019d)

Kuvassa 3 on esimerkki Wit.ai:n palauttamasta JSON vastauksesta, kun käyttäjä esimerkiksi pyytä Wit.ai:ta etsimään asunnon, jonka sijainti on Palo Alto Kaliforniassa. Vastauksena saatava viesti sisältää tiedon siitä, minkä tyylisen kysymyksen käyttäjä on esittänyt

ja erilaista oleellista tietoa liittyen vastaukseen. Viesti sisältää myös tiedon siitä, kuinka varma Wit.ai on vastauksen oikeellisuudesta. Kehittäjä voi hyödyntää Wit.ai:n varmuus prosenttia päättelemään onko saatu vastaus luotettava vai ei. Epävarman vastauksen kohdalla sovellus voi pyytää tarvittaessa käyttäjää tarkentamaan kysymystä. (Wit.ai.2019d.)

Wit.ai sisältää useita sisäänrakennettuja entiteettejä. Nämä ovat usein käytettyjen pyyntöjen kuten lämpötilan tai päivämäärän kyselyt tai esimerkiksi tervehdysten tunnistaminen. Sisäänrakennetut entiteetit tunnistetaan siitä, että niiden alussa on ”wit/” merkintä.

- wit/contact
- wit/datetime
- wit/distance
- wit/greetings
- wit/location.

(Wit.ai 2019d.)

Yllä olevassa listassa on esimerkkejä Wit.ai:n sisältämistä valmiista entiteeteistä. Wit/contact entiteetti mahdollistaa nimien tai henkilöviittausten tunnistamisen. Henkilöviittausten avulla voi esimerkiksi tunnistaa, jos lauseessa puhutaan lääkäristä tai viitataan henkilöihin nimellä. Wit/datetime mahdollistaa esimerkiksi kysymyksen siitä, milloin on seuraava viikonloppu tai mikä päivä on huomenna. Wit/distance mahdollistaa kyselyt kahden paikan väliltä, esimerkiksi kuinka pitkä matka on Pariisista Helsinkiin.

Yhdistämällä etäisyyden pyytämisen muihin Wit.ai:n ominaisuuksiin, voi esimerkiksi laskea etäisyyden lähimpään kauppaan tai ravintolaan. Wit/greetings tunnistaa englannin kielisiä tervehdyksiä kuten ”hello” tai ”how are you”. Näitä voi hyödyntää esimerkiksi komentoina, joihin keskustelurobotti reagoi ja aloittaa sille ohjelmoidun keskustelun käyttäjän kanssa. Wit/location mahdollistaa paikkakuntien tai osoitteiden tunnistamisen lauseista. Tätä voi hyödyntää esimerkiksi koordinaattien pyytämiseen. Osoitteiden pyytäminen on myös hyödyllistä yhdistää esimerkiksi ajo-ohjeiden etsimiseen. (Wit.ai 2019d.)

5 KÄYTÄNNÖN TOTEUTUS

5.1 Toimeksiantaja

Opinnäytetyö toteutettiin lahtelaiselle Avenla Oy:lle. Yritys on Lahdessa, Helsingissä ja Tampereella toimiva digitaalisia palveluita ja työkaluja tarjoava yritys. Organisaatioon kuuluu yli 20 henkeä. Avenla Oy tarjoaa asiakkailleen jatkuvasti kehittyviä ja ketteriä työkaluja sekä alustoja, jotka räätälöidään asiakaskohtaisesti tehostamaan asiakkaan digitaalista toimintaa. Palveluihin kuuluu asiakkaan sähköisen markkinoinnin ja liiketoiminnan tehostaminen asiakaskohtaisilla ratkaisuilla. (Avenla Oy 2019.) Toimeksianto suoritettiin Avenla Oy:n ohjeistuksen mukaisesti itsenäisenä testinä, jonka tarkoituksena on havaita erilaisia mahdollisuuksia jatkokehitystä varten.

5.2 Soveltuvuusselvityksen esittely

Tavoitteena oli testata puheensyntetisoinnin ja puheentunnistuksen käyttö mahdollisuuksia ja niiden hyödyntämistä erilaisten taustajärjestelmien kanssa. Puheentunnistuksen ja syntetisoinnin toteutukseen käytettiin Googlen ilmaista puheentunnistus rajapintaa. Googlen rajapinta valittiin käytettäväksi sen helpon saatavuuden takia. Rajapinta toimii Javascriptillä ja on sisällytettynä Google Chrome nettiselaimeen ja on osittain tuettuna muissakin suurissa verkkoselaimissa. Rajapinnan käyttöönotto ja liikkeelle pääseminen on nopeaa ja vaivatonta, koska se ei vaadi sovellusten asentamista tai rekisteröitymistä.

Kehitettävän sovelluksen taustajärjestelmänä käytetään Facebookin omistamaa Wit.ai NLP alustaa, jonka avulla tunnistetulle puheelle saadaan eriteltyä merkitys.

5.3 Puheensyntetisoinnin toteutus

Ensimmäinen askel puheentunnistus ja syntetisointisovelluksen kehityksessä on luoda Javascript tiedosto, jonka sisälle koko sovellus kirjoitetaan. Javascript tiedoston luonnin jälkeen sovelluksen kehityksessä nopein tapa lähteä liikkeelle on aloittaa puheensyntetisoinnilla. Puheensyntetisointi on helpompaa kuin puheen tunnistus, sillä syntetisoinnissa sovellukselle voi antaa testaamista varten minkä vain teksti syötteen, jonka sovellus voi lukea.

```
var msg = new SpeechSynthesisUtterance('Hello World');
window.speechSynthesis.speak(msg);
```

KUVA 4. Puheentunnistuksen testaamiseen käytettävä Javascript koodi (Bidelman 2019)

Kuvassa 4 on lyhyt Javascript koodi, jonka avulla Google Chrome lukee ääneen sanat ”Hello World”. Kuva osoittaa, kuinka nopeasti puheensyntetisoinnissa on mahdollista päästä liikkeelle ja tunnistaa mahdollisia ongelmia jo kehityksen alkuvaiheessa.

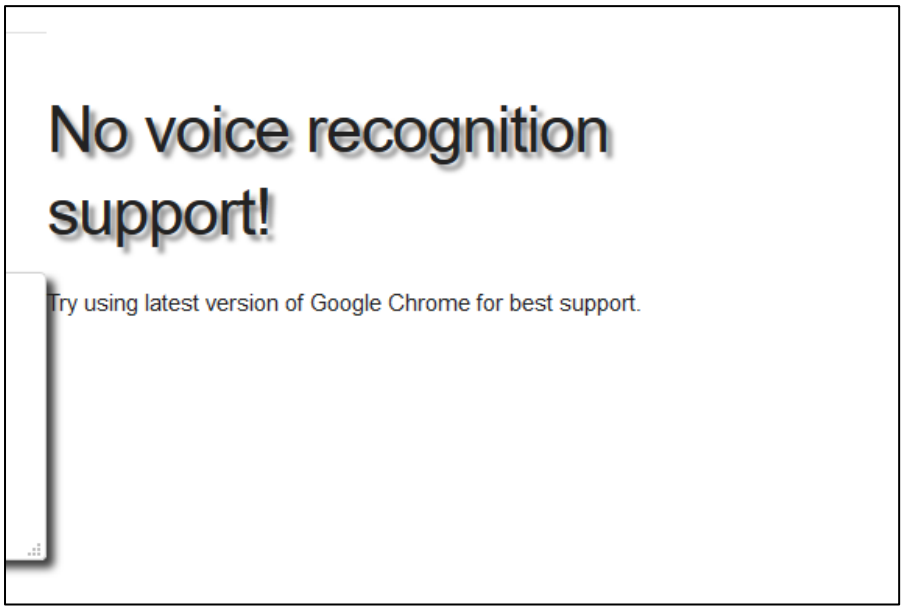
Yksinkertaisen puheensyntetisointi testin jälkeen kannattaa huomioida eri internet selainten tuki puheentunnistuksessa ja syntetisoinnissa käytettävälle Webkit-Speech-Recognition komponentille. Komponentin tuki vaihtelee selaimittain ja käyttäjän kannalta on tärkeää, ettei sovellus yritä toimia ilman kunnollista tukea tarvittaville toiminnoille. Webkit-Speech-Recognition komponentin tuen tarkistaminen onnistuu Javascript koodissa if else rakenteen avulla.

```
function CheckSynthesisAndRecognitionSupport() {
    if ('speechSynthesis' in window) {
        $('#VoiceSynthesis').show();
        $('#NoSynthesis').hide();
    }
    else {
        $('#VoiceSynthesis').hide();
        $('#NoSynthesis').show();
    }

    if (('webkitSpeechRecognition' in window)) {
        $('#VoiceRecognition').show();
        $('#NoRecognition').hide();
    }
    else {
        $('#VoiceRecognition').hide();
        $('#NoRecognition').show();
    }
    $('#CreateAlertMessage').hide();
    $('#AddAlertComments').hide();
}
```

KUVA 5. Javascript koodi Webkit-Speech-Recognition komponentin tuen tarkistamiseen

Kuvassa 5 on funktio, jota kutsutaan sivun latauksen yhteydessä, joka tarkistaa tukeeko käyttäjän selain tarvittavaa komponenttia. Selaimet voivat tukea Webkit-Speech-Recognition komponenttia myös osittain. Esimerkiksi Mozilla Firefox ja Microsoft Edge selaimet tukevat komponentista ainoastaan puheen syntetisointia, kun taas Google Chrome tukee myös puheen tunnistusta. Kuvan 4 koodi tarkistaa erikseen tuen sekä puheentunnistukselle että syntetisoinnille. Mikäli komponentin tuki puuttuu, käyttäjälle näytetään selaimessa ilmoitus puuttuvasta tuesta ja suositus kokeilla uusinta versiota Google Chromesta parhaan tuen saamiseksi.



KUVA 6. Käyttäjälle esitettävä viesti puuttuvasta tuesta Webkit-Speech-Recognition puheentunnistukselle

Kuvassa 6 näkyy käyttäjälle esitettävä ilmoitus Firefox-selainta käytettäessä. Sivun lataamisen yhteydessä suoritettu tarkistus puheensyntetisoinnille- ja tunnistukselle varmistaa sen, ettei käyttäjä joudu tilanteeseen, jossa sivu kaatuu virheisiin, vaan käyttäjä saa heti ilmoituksen puuttuvasta tuesta.

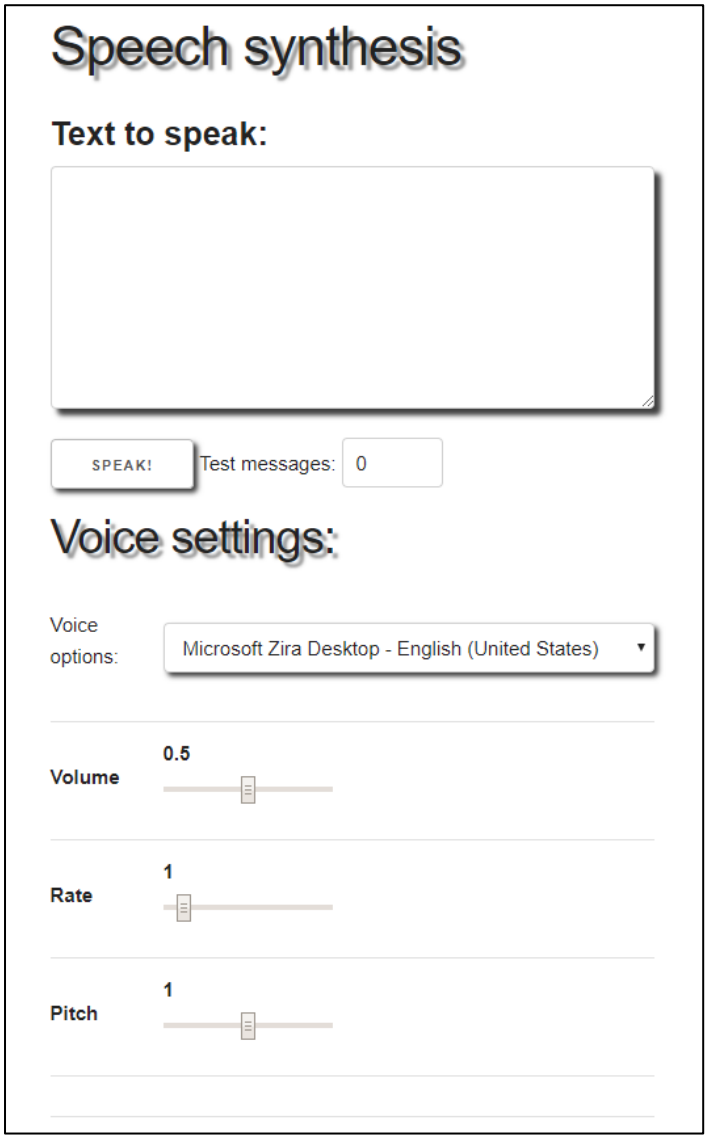
Puheensyntetisoinnin testaamisen jälkeen seuraavana on suositeltavaa parannella puheensyntetisointiin liittyvää koodia. Puheentunnistuksen testaamiseen käytettävä ”Hello World” teksti korvataan muuttujalla, jolloin sovellukselle voidaan määrittää käytön aikana vaihtuvia syntetisoitavia sanoja tai lauseita. Syntetisoitavan tekstin lisäksi Webkit-Speech-Recognition komponentti mahdollistaa useiden erilaisten asetusten muuttamisen ja säätämisen, joiden avulla puheensyntetisoinnista saa paremmin toimivan ja eri kielille sopivan.

```
97   function Speak(message) {
98       CancelSynthesis();
99       var voice = SetSpeechValues();
100      var msg = new SpeechSynthesisUtterance();
101      var voices = window.speechSynthesis.getVoices();
102
103      if ($('#VoiceOptions option').length)
104          msg.voice = voices[$('#VoiceOptions').val()];
105
106      msg.voiceURI = 'native';
107      msg.volume = voice["volume"]; // 0 to 1
108      msg.rate = voice["rate"]; // 0.1 to 10
109      msg.pitch = voice["pitch"]; //0 to 2
110      msg.text = message;
111      msg.lang = 'en-US';
112
113      speechSynthesis.speak(msg);
114  }
115
```

KUVA 7. Puheensyntetisointiin käytettävä funktio

Kuvassa 7 on funktio, jota kutsumalla sovellus syntetisoi puhetta ajon aikana. Funktiolle täytyy kutsun yhteydessä välittää syntetisoitava viesti. Funktion sisällä puheensyntetisoinnille määritellään muitakin asetuksia kuin pelkkä sisältö. Puheensyntetisointiin käytetylle komponentille määritellään muun muassa äänen voimakkuus, nopeus ja korkeus. Näiden avulla voidaan vaikuttaa siihen, miltä syntetisoitava viesti kuulostaa ja kuinka nopeasti se luetaan. Lisäksi funktiossa määritellään ääni, jota syntetisointiin käytetään.

Puheensyntetisoinnin asetuksissa täytyy ottaa huomioon eri asetusten äärirajat. Äänenvoimakkuuden arvo tulee olla desimaaliluku arvojen 0 ja 1 väliltä. Syntetisoitavan äänen nopeus tulee olla desimaaliluku 0.1 ja 10 välillä ja äänen korkeuden tulee olla väliltä 0 ja 2. Näiden arvojen avulla käyttäjä pystyy muuttamaan ääntä sopivammaksi tilanteisiin, koska osa syntetisointiin käytettävistä äänistä saattaa aksentin takia olla oletuksena hieman erikuuloisia.



KUVA 8. Puheensyntetisoinnin asetukset sovelluksessa

Kuvassa 8 näkyy sovelluksen käyttöliittymässä olevat asetukset, joiden avulla käyttäjän on mahdollista vaikuttaa puheensyntetisointiin. Käyttäjän vaihtaessa syntetisointiin liittyvää ääntä, vaihtuu samalla myös äänen aksentti ja kieli. Tämä tarkoittaa sitä, että mikäli käyttäjän kirjoittama teksti on suomenkielinen, kannattaa valita syntetisoinnin kieleksi suomi. Väärän kielen käyttäminen syntetisoinnissa tekee äänestä hankalan ymmärtää.

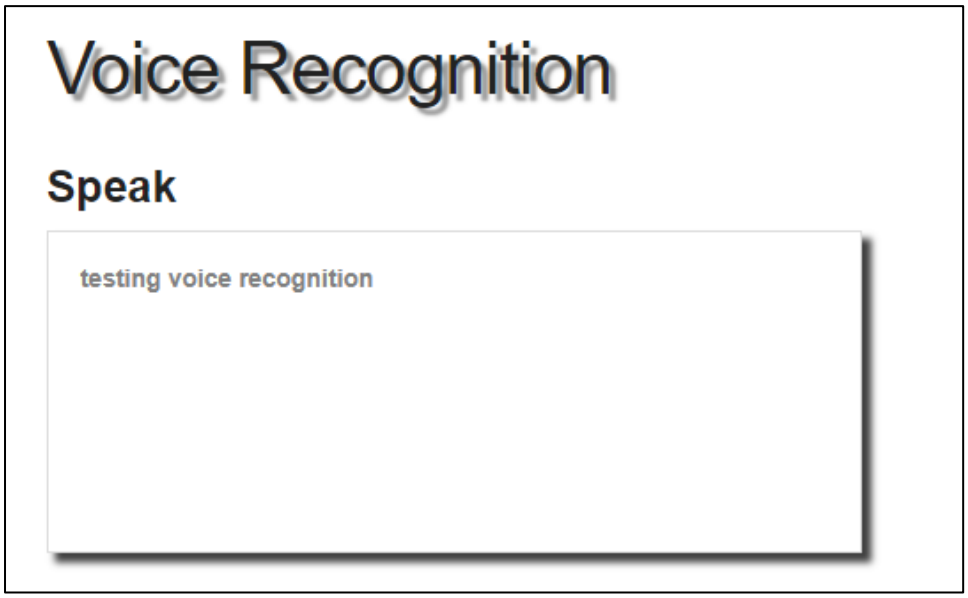
Sovelluksen tukemat kielivalinnat ovat oletuksena selainkohtaisia, mikä tarkoittaa sitä, ettei sovelluksen kehittäjä pysty täysin määrittelemään mitä kieliä syntetisointi tukee. Sovelluskehittäjä pystyy kuitenkin hyödyntämään myös käyttäjän laitteelle asennettuja kieliä. Laitteen kielien käyttäminen vaatii kuitenkin käyttäjältä toimenpiteitä, koska oletuksena suurin osa laitteista sisältää vain yhden kielen.

5.4 Puheentunnistus

Puheensyntetisoinnin jälkeen seuraava askel on toteuttaa puheentunnistus sovellukseen. Puheentunnistus on kehittäjän kannalta monimutkaisempi saada toimimaan kuin puheensyntetisointi. Puheentunnistuksessa on tärkeää pitää huoli siitä, että käyttäjän selain tukee Webkit-Speech-Recognition komponenttia. Kehittäjä voi olla varma ainoastaan siitä, että Google Chrome selain tukee puheentunnistusta, joten puheentunnistuksen tuen tarkistaminen ennen sen käyttöä on erityisen tärkeää.

Puheentunnistuksessa kehittäjälle sekä käyttäjällä ei ole yhtä paljon asetuksia käytettävissä kuin puheensyntetisoinnissa, mutta puheentunnistuksen asetukset ovat tärkeitä sovelluksen hyvän käyttökokemuksen takia. Kehittäjä voi valita sovelluksen käyttökohteen mukaan toimiiko puheentunnistus jatkuvana vai niin, että se katkeaa, kun käyttäjä lopettaa puhumisen. Jatkuvan tunnistuksen asetus ei oletuksena ole päällä, joten mikäli käyttäjä tahtoo hyödyntää sitä, on se erikseen laitettava päälle.

Tutkimuksessa toteutettu sovellus antaa käyttäjälle mahdollisuuden valita haluaako käyttäjä jatkuvan tunnistuksen olevan käytössä vai ei. Toinen asetus, jolla kehittäjä voi vaikuttaa sovelluksen käyttäjän kokemukseen on väliaikaisten puheentunnistutulosten näyttäminen. Kehittäjä voi valita näyttääkö sovellus reaaliaikaisesti käyttäjälle, mitä sovellus on tunnistanut vai asettaako sovellus tuloksen vasta lopuksi. Tutkimuksessa toteutettu sovellus näyttää käyttäjälle tulokset reaaliaikaisesti. Tämä helpottaa käyttö kokemusta, koska käyttäjä näkee välittömästi sovelluksen toimivan ja tunnistavan puhetta.



KUVA 9. Puheentunnistuksen tulokset

Kuvassa 9 näkyy puheentunnistuksen väliaikainen tulos. Väliaikaisen tuloksen näyttäminen parantaa sovelluksen käyttökokemusta, koska käyttäjä näkee jatkuvasti mitä sovellus tekee ja tunnistaako sovellus puhetta oikein. Väliaikainen tunnistuksen tulos on kuitenkin vain suuntaa antava ja ennen lopullisen tuloksen asettamista, se voi vielä muuttua.

```
454  if (!('webkitSpeechRecognition' in window)) {
455      upgrade();
456  } else {
457      var recognition = new webkitSpeechRecognition();
458      recognition.continuous = true;
459      recognition.interimResults = true;
460
461      recognition.onstart = function () {
462          RecognitionOnStart();
463      };
464
465      recognition.onerror = function (event) {
466          RecognitionOnError();
467      };
468
469      recognition.onend = function () {
470          RecognitionOnEnd();
471      };
472
473      recognition.onresult = function (event) {
474          RecognitionOnResult();
475      };
476  }
```

KUVA 10. Puheen tunnistuksen asetukset koodissa)

Kuvassa 10 on Javascript koodi, joka ensin varmistaa, että käyttäjän selain tukee puheentunnistusta ja tämän jälkeen luo puheentunnistus komponentin. Tämän jälkeen koodissa asetetaan sovelluskohtaiset oletusarvot puheentunnistuksen jatkuvuudelle ja väliaikaisten tulosten näyttämiselle. Tutkimuksessa toteutettu sovellus antaa käyttäjälle mahdollisuuden vaikuttaa käytön aikana siihen, toimiiko puheentunnistus jatkuvana vai ei, mutta oletuksena puheentunnistus toimii jatkuvana. Oletusarvojen asettamisen jälkeen asetetaan

vielä toiminnot puheentunnistuksen aloitukselle, lopetukselle, virhetilanteille sekä tulosten saamiselle.

```
242  function StartSpeechRecognition() {
243      if (recognizing) {
244          recognition.stop();
245          return;
246      }
247      final_transcript = '';
248      recognition.lang = select_dialect.value;
249      recognition.start();
250      ignore_onend = false;
251      final_span.innerHTML = '';
252      interim_span.innerHTML = '';
253      SetInfoText("Starting voice recognition");
254      start_timestamp = event.timeStamp;
255      ToggleButtonVisibilities();
256      console.log("Starting");
257  }
```

KUVA 11. Puheentunnistuksen aloittavan napin koodi

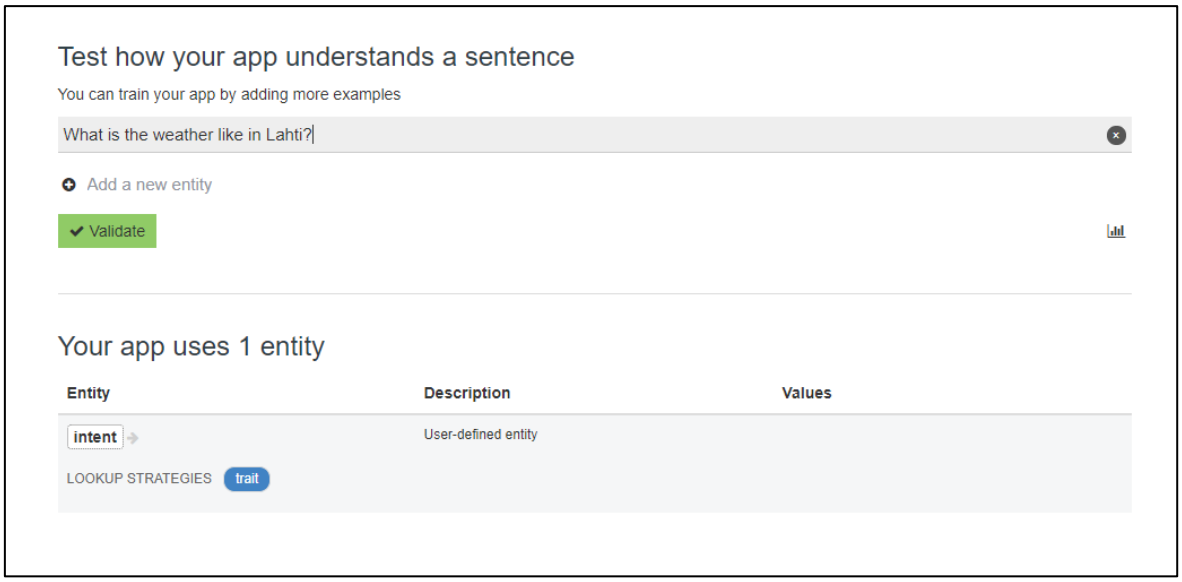
Kuvassa 11 on koodi, joka suoritetaan, kun käyttäjä painaa sovelluksen käyttöliittymässä olevaa nappia puheentunnistuksen aloittamiseksi. Sama koodi toimii sekä puheentunnistuksen aloittamiseen ja lopettamiseen. Ensimmäisenä koodissa tarkistetaan, mikä on puheentunnistuksen tilanne napinpainamisen hetkellä. Mikäli puheentunnistus on jo käynnissä, asetetaan puheentunnistuksen tilaksi pysäytetty. Puheentunnistuksen tilan vaihtuessa Webkit-Speech-Recognition komponentti havaitsee puheentunnistuksen lopetukseen liittyvän tapahtuman ja suorittaa kuvassa 10 määritetyn ”recognition.onend” funktion, joka pysäyttää puheentunnistuksen. Mikäli puheentunnistus ei vielä ole käynnissä, napin painaminen käynnistää puheentunnistuksen ja tyhjentää edellisen puheentunnistuksen tulokset, sekä vaihtaa aloitusnapin tekstin ja näyttää käyttäjälle infotekstin siitä, että puheentunnistus on käynnissä.

KUVA 12. Käyttäjälle näytetty info teksti ja mikrofoni ikoni

Kuvassa 12 näkyy puheentunnistuksen käyttöliittymä, kun puheentunnistus on käynnissä. Kuvan yläreunassa oleva ”Speak” teksti näyttää käyttäjälle ohjeita ja tietoa puheentunnistuksen tilanteesta. Lopetusnapin vieressä näkyy punainen mikrofonin kuva, joka ilmaisee käyttäjälle puheentunnistuksen olevan käynnissä. Puheentunnistuksen loppuessa mikrofonin kuva häviää kokonaan näkyvistä.

5.5 Wit.ai:n kouluttaminen

Puheentunnistuksen ja syntetisoinnin jälkeen seuraavana valmistellaan Wit.ai NLP alusta käyttöä varten. Wit.ai:n käyttöönoton aloittaminen alkaa sisäänkirjautumisella ja uuden projektin luomisella. Uuden projektin luomisen jälkeen Wit.ai tekoälyä täytyy kouluttaa sovelluksen tarpeisiin sopivaksi. Wit.ai:n kouluttaminen tarkoittaa sitä, että Wit.ai:n käyttöliittymässä sovellukselle annetaan esimerkki lauseita, joita sovellukseen oletettavasti tulee normaalin käytön aikana. Kouluttamisen avulla Wit.ai tekoäly opetetaan erottelemaan esimerkkilauseiden kaltaisista lauseista niiden merkitykset ja entiteetit



KUVA 13. Esimerkki lauseen opettaminen Wit.ai:lle

Kuvassa 13 näkyy Wit.ai:n käyttöliittymä, jossa sovelluksen koulutus tapahtuu. Kouluttamisen alussa sovellukselle syötetään esimerkki lause, jonka kaltaisia lauseita sovelluksen loppukäyttäjä voi sovellukselle esittää. Kuvassa 13 sovellukselta tiedustellaan englannin kielellä säätä Lahden kaupungissa.

Kuva 14. Merkityksen ja entiteetin opettaminen Wit.ai:lle

Kuvassa 14 opetetaan Wit.ai:lle tunnistamaan merkitys ja entiteetti kuvan 13 esimerkkilauseesta. Esimerkkilauseen kohdalla merkitys on sään kysyminen ja entiteetti on haluttu paikkakunta. Esimerkkilauseen luonnin yhteydessä luodaan uusi merkitys sään hakemista varten ja annetaan sille kuvaava nimi. Tutkimuksessa käytetyssä sovelluksessa merkityksen nimeksi asetettiin ”temperature_get”.

Säätietojen kysymiseen liittyy myös entiteetti, joka esimerkkilauseen yhteydessä on Lahti. Wit.ai sisältää valmiin entiteetin paikkakuntien ja sijaintien tunnistamiseen, joten on järkevää käyttää valmiiksi määriteltyä wit/location entiteettiä. Entiteetin määrittämisen

yhteydessä käyttöliittymässä täytyy valita hiirellä maalaamalla paikkakunta. Tämän valinnan avulla Wit.ai pystyy tunnistamaan, mitä paikkakuntaa säätietojen hakeminen koskee. Wit.ai näyttää merkitysten ja entiteettien vieressä varmuusarvon. Tämä arvo on desimaaliluku 0:n ja 1:n välissä ja kertoo Wit.ai:n varmuuden siitä, että se on tunnistanut merkityksen ja entiteetin oikein.


```
{
  "_text": "How is the weather like in Lahti?",
  "entities": {
    "location": [
      {
        "confidence": 0.93409,
        "value": "Lahti",
        "resolved": {
          "values": [
            {
              "name": "Lahti",
              "grain": "locality",
              "type": "resolved",
              "timezone": "Europe/Helsinki",
              "coords": {
                "lat": 60.982669830322,
                "long": 25.661510467529
              },
              "external": {
                "geonames": "649360",
                "wikidata": "Q28911",
                "wikipedia": "Lahti"
              }
            },
            {
              "name": "Lahti",
              "grain": "region",
              "type": "resolved",
              "timezone": "Europe/Helsinki",
              "coords": {
                "lat": 60.982769012451,
                "long": 25.654109954834
              },
              "external": {
                "geonames": "649374",
                "wikidata": "Q33688990"
              }
            }
          ]
        }
      }
    ],
    "intent": [
      {
        "confidence": 0.99900312661497,
        "value": "temperature_get"
      }
    ]
  },
  "msg_id": "1VUKtFuI8xt8X5Jfy"
}
```


KUVA 15. Wit.ai:n esimerkkilauseen mallivastaus

Kuvassa 15 näkyy Wit.ai:n luoma JSON muotoinen esimerkkivastaus. Kehittäjä voi tarkistaa esimerkkilauseen luonnin yhteydessä minkälaisen vastauksen Wit.ai antaa luodulle lauseelle. Esimerkkivastauksen saa tarkistettua kuvan 14 oikeassa alareunassa olevasta painikkeesta, joka kuvaa tilastoja.

Esimerkkilauseen merkityksen ja entiteettien valinnan jälkeen esimerkki hyväksytään painamalla Validate nappia käyttöliittymässä. Wit.ai pystyy hyväksymisen jälkeen käyttämään esimerkkilausetta vertaamaan normaalin käytön aikana saamiaan lauseita esimerkkeihin ja pyrkii tunnistamaan yhtäläisyyksiä niiden kanssa. Wit.ai:n tarkkuuden kannalta on

tärkeää luoda useita esimerkkilauseita, jotka vaihtelevat hieman sanamuodoittain. Mitä enemmän esimerkkejä Wit.ai:lle antaa, sen tarkemmin se pystyy käytönaikana toimimaan.

Kehittäjän näkökulmasta esimerkkilauseiden luonnin jälkeen Wit.ai on käyttövalmis. Wit.ai on kuitenkin tekoälyalusta, joten pelkät esimerkkilauseet eivät riitä, vaan se vaatii myös aikaa, jotta se oppii tunnistamaan lauseet oikein. Wit.ai sisältää kuitenkin automaattisen ajastetun koulutusominaisuuden, joten kehittäjän ei tarvitse huolehtia muusta kuin esimerkkilauseista.

5.6 Kommunikointi Wit.ai:n kanssa

Wit.ai tekoälyn kouluttamisen jälkeen toteutettava sovellus täytyy saada kommunikoimaan Wit.ai:n kanssa. Viestien lähettämisen lisäksi Wit.ai:n vastaanottamat viestit täytyy myös hyväksyä, jotta NLP-sovellus ei opi vääriä asioita.

5.6.1 HTTP-kutsun tekeminen

Kommunikaatio kehitettävän sovelluksen ja Wit.ai:n välillä tapahtuu HTTP pyyntöjen avulla.

```
149  function WitApiCall() {
150      let messageToWit = "";
151      if (resultsList[resultsList.length - 1])
152          messageToWit = resultsList[resultsList.length - 1];
153      console.log("Message: " + messageToWit);
154      $.ajax({
155          url: 'https://api.wit.ai/message',
156          data: {
157              'q': messageToWit,
158              'access_token': 'TKFERTECVYUCOQY3WZMZFAKGS4WYW66S'
159          },
160          dataType: 'jsonp',
161          method: "GET"
162      })
163      .done(function (response) {
164          FindActions(response);
165      })
166      .fail(function (xhr) {
167          console.log('error', xhr);
168      });
169  }
170
```

KUVA 16. HTTP pyyntö Wit.ai:lle

Kuvassa 16 näkyy Javascript funktio, joka lähettää HTTP pyynnön Wit.ai tekoälylle analysoitavaksi. Funktiossa tallennetaan ”messageToWit” muuttujaan viimeisin viesti, joka sovelluksen lokiin on kirjoitettu. Tämä muuttuja lähetetään HTTP pyynnön mukana Wit.ai:lle. Tekstimuuttujan lisäksi pyynnön mukana lähetetään myös sovelluskohtainen avain, jota

käytetään sovelluksen todentamiseen. HTTP kutsun lähettämiseen käytettävä funktio on toteutettu vikasietoiseksi, jolloin epäonnistunut HTTP pyyntö ei aiheuta sovelluksen kaatumista, vaan toteuttaa hallitusti sille määritetyn virheilmoituksen.

Onnistuneen HTTP pyynnön jälkeen sovellus odottaa JSON muotoista vastausta Wit.ai:lta, jonka sovellus sitten välittää toiselle funktiolle merkityksen ja entiteettien erittelyä varten.

```
function FindActions(response) {
    let parameters = [];
    try {
        let intent = response["entities"]["intent"];
        if (intent)
            parameters.push(intent[0]["value"]);
        for (let key in response["entities"]) {
            if (response["entities"].hasOwnProperty(key)) {
                let val = response["entities"][key];
                if (key == "greetings") {
                    parameters.push(key);
                }
                else if (val[0]["value"]) {
                    if (parameters.indexOf(val[0]["value"]) == -1)
                        parameters.push(val[0]["value"]);
                }
            }
        }
        predefinedFunctions.ChooseAction(parameters);
    }
    catch (e) {
        predefinedFunctions.ChooseAction(parameters);
        hasSetResultsAlready = false;
    }
}
```

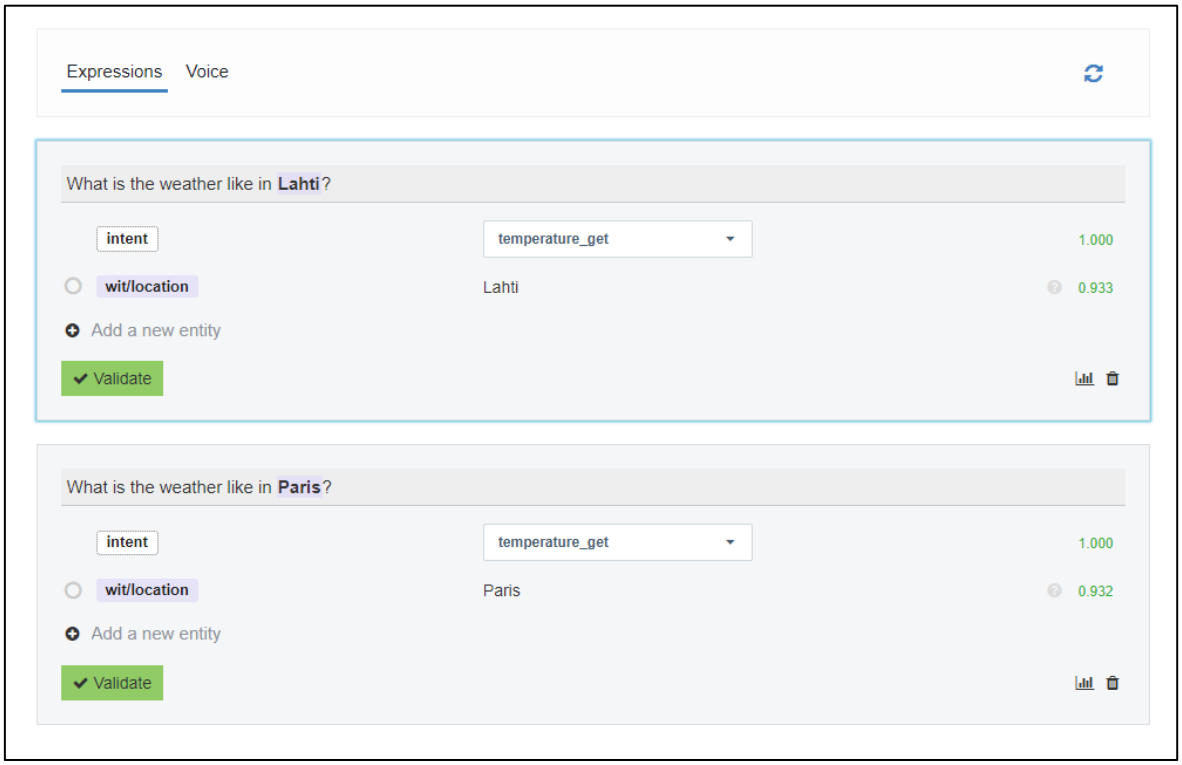
KUVA 17. Merkityksen ja entiteetin erittelyyn käytetty funktio

Kuvassa 17 on funktio, jota käytettään Wit.ai:n palauttaman JSON viestin merkityksen ja entiteettien erittelyyn. Funktio etsii ensimmäisenä JSON viestistä merkityksen ja lisää sen parameters-listaan. Merkityksen etsimisen jälkeen funktio etsii viestin entiteetit ja lisää ne yksi kerrallaan merkityksen perään parameters-listaan. Lopulta viesti lähetetään erilliseen funktioon, joka toteuttaa merkityksen ja entiteettien perusteella tarvittavat toiminnot.

5.6.2 Tulosten hyväksyminen Wit.ai:ssa

Kehittäjä voi sovelluksen käytön aikana käydä tarkistamassa Wit.ai hallintapaneelista mitä viestejä Wit.ai on vastaanottanut ja miten se on niitä luokitellut. Hallintapaneelista on mahdollista muun muassa poistaa viestejä, jotka sovellus on vastaanottanut, mutta joita sovelluksen ei ole tarkoitus ymmärtää. Tämän lisäksi kehittäjä voi muuttaa Wit.ai:n luokituksia, mikäli ne eivät ole oikein. Sovelluksen käytön aikana saamat viestit kehittävät Wit.ai:ta

toimimaan entistä paremmin, joten näitä viestejä kannattaa käydä varsinkin alkuvaiheessa merkitsemässä usein.



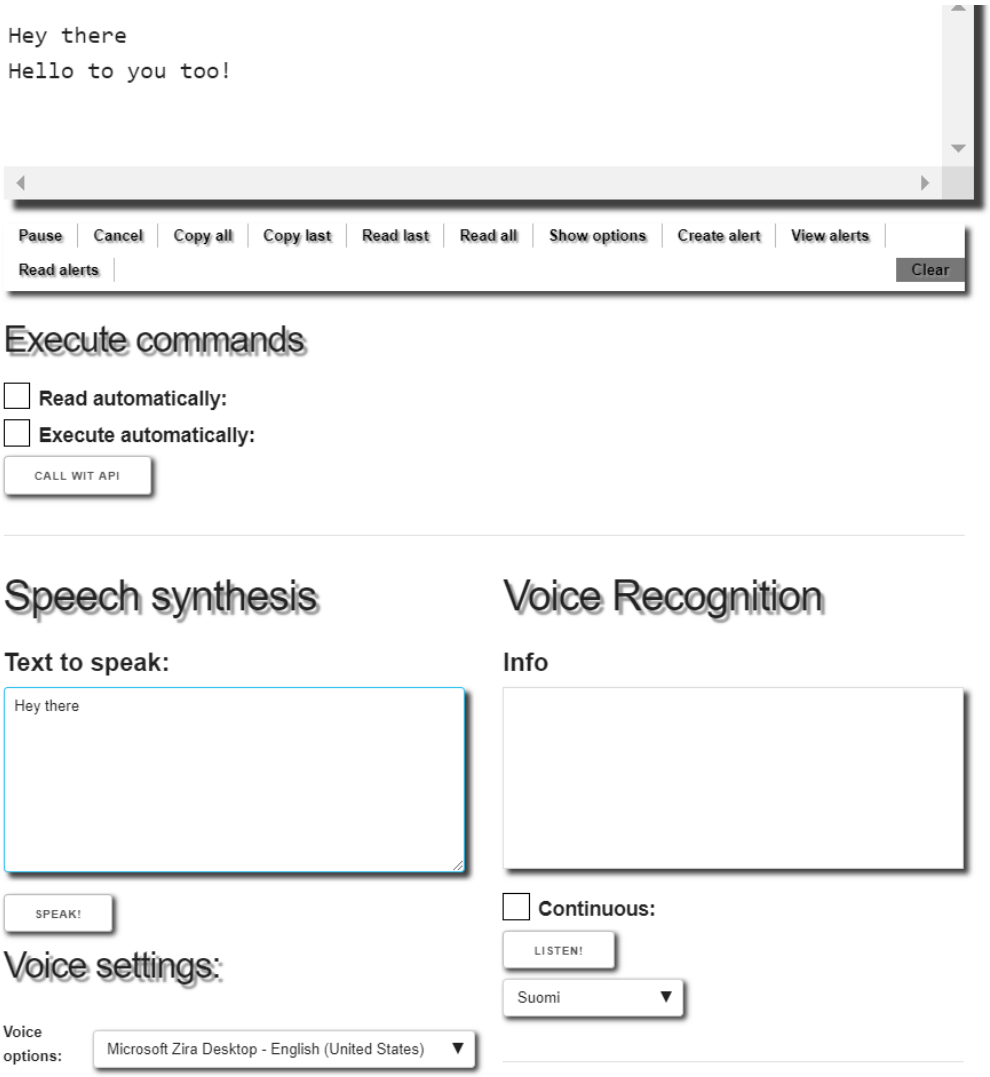
KUVA 18. Tulosten hyväksyminen Wit.ai hallintapaneelissa

Kuvassa 18 näkyy sovelluksen käyttäjän Wit.ai:lle lähettämät kysymykset hallintapaneelissa. Wit.ai tunnistaa, että molemmat kysymykset koskevat säätietoja ja osaa tunnistaa käyttäjän pyytämät säätiedot paikkakunnille. Käyttäjä voi myös esittää kysymyksen hieman eri tavalla, mutta Wit.ai tunnistaa silti, mitä käyttäjä tarkoittaa.

Wit.ai ei hyödynnä käyttäjän sille lähettämiä eri muodossa esitettyjä viestejä oletuksena, vaan kehittäjän täytyy käydä ne erikseen vahvistamassa. Tämä estää NLP prosessin kehittymisen virheelliseen tai kehittäjän kannalta epäsuotuisaan suuntaan. Wit.ai ei myöskään vaadi kehittäjää vahvistamaan samoja viestejä useaan kertaan, vaan jos viesti on jo kerran hyväksytty, niin Wit.ai osaa hyödyntää sitä automaattisesti jatkossa.

5.6.3 Käyttöliittymän toteutus

Perusominaisuuksien kehittämisen ja tekoälyn kouluttamisen jälkeen viimeistellään sovelluksen käyttöliittymä. Käyttöliittymään tulee aiemmin luodut erilliset puheentunnistus ja syntetisointi näkymät, sekä erillinen lokinäkymä. Lokinäkymässä näkyy käyttäjän sovellukseen syöttämät viestit ja tekoälyn antamat vastaukset.



KUVA 19. Sovelluksen käyttöliittymä

Kuvassa 19 näkyy sovelluksen käyttöliittymä, joka koostuu kolmesta osasta. Käyttöliittymän yläreunassa on lokinäkymä, jonne käyttäjän viestit ja tekoälyn vastaukset tulevat näkyviin. Käyttäjä voi hyödyntää sovelluksen ominaisuuksia lokin kopiointiin tai tyhjentämiseen. Lokin alla on puheentunnistus ja -syntetisointi näkymät. Käyttäjä voi hyödyntää sekä puheentunnistusta että syntetisointia sovelluksen ohjaamiseen. Jokaisen eri näkymän alla on kyseiseen näkymään liittyvät asetukset, joiden avulla käyttäjä voi säätää sovelluksen toimintaa.

5.7 Sovelluksen ennalta määrätyt komennot

NLP alustat kuten Wit.ai eivät tarjoa vastauksia käyttäjän kysymyksiin, vaan ne ainoastaan erittelevät saamansa viestin merkityksen ja entiteetit. Tämän seurauksena kehittäjän on pidettävä huoli, että kehitetty sovellus osaa reagoida Wit.ai:n palauttamiin JSON viesteihin.

Viesteihin reagoimista varten sovellukseen kehitettiin erillinen Javascript luokka ennalta määrätyille komennoille. Tämä Javascript luokka vertaa Wit.ai:n palauttamaa merkitystä ja entiteettejä sovellukseen ohjelmoituihin ennalta määrättyihin komentoihin ja joko suorittaa niitä vastaavat toimenpiteet tai ilmoittaa käyttäjälle, ettei sovellus tunnista sille annettua käskyä. Komentojen tunnistusta varten luodussa luokassa on funktio, joka vastaanottaa listan merkityksistä ja entiteeteistä. Tunnistukseen käytetty funktio olettaa, että viestin merkitys on asetettu listan ensimmäiseksi alkioksi, jotta merkitys ja entiteetit on helppo käsitellä erikseen.

```
17  ChooseAction(actionType) {
18      switch (actionType[0]) {
19          case "greetings":
20              this.GreetUser();
21              break;
22          case "temperature_get":
23              this.GetWeather(actionType[1]);
24              break;
25          case "results_clear":
26              this.ClearResults();
27              break;
28          case "messages_last":
29              this.StartExecution();
30              this.ReadLastMessage();
31              break;
32          case "messages_all":
33              this.StartExecution();
34              this.ReadAllMessage();
35              break;
36          case "copy_last":
37              this.StartExecution();
38              this.CopyLastMessageToClipboard();
39              break;
40          case "copy_all":
41              this.StartExecution();
42              this.CopyToClipboard();
43              break;
```

Kuva 20. Ennalta määriteltyjä komentoja käsittelevä funktio

Kuvassa 20 on funktio, joka vertaa saamaansa listaa viestin merkityksestä ja entiteeteistä sovellukseen määriteltyihin komentoihin ja suorittaa halutun toiminnon. Funktio hyödyntää Javascriptin Switch Case rakennetta oikean toiminnon löytämiseen. Osa sovelluksen tukemista ominaisuuksista toimii pelkän merkityksen avulla, mutta esimerkiksi säätietojen pyytäminen vaatii myös entiteetin, jotta säätiedot voidaan hakea oikealle paikkakunnalle.

Sovelluksen tukemia ennalta määrättyjä komentoja ovat:

- Käyttäjän tervehtiminen
- Säätietojen kysyminen
- Lokin kopiointi leikepöydälle
- Viimeisen viestin kopioiminen leikepöydälle

- Lokin lukeminen ääneen
- Viimeisen viestin lukeminen ääneen
- Puheentunnistuksen jatkuvuuden valinta
- Ennalta määrättyjen toimintojen tulostaminen
- Hälytysten luominen
- Hälytysten kuittaaminen

Mikäli käyttäjä yrittää suorittaa toimintoa, jota ei ole määritelty, sovellus ilmoittaa käyttäjälle, ettei kyseistä komentoa voi suorittaa.

6 YHTEENVETO

Työn tavoitteena oli suorittaa soveltuvuusselvitys puheentunnistuksesta ja syntetisoinnista. Tarkoituksena oli testata valmiiden puheentunnistuskomponenttien käyttöä osana sovellusta, joka myöhemmin voisi hyödyntää näitä ominaisuuksia ja kytkeytyä laajempaan taustajärjestelmään.

Puheentunnistuksen kehittyminen on nykyisin voimakasta ja yksi iso syy tälle kehitykselle on sekä laitteiden että erilaisten puheentunnistus rajapintojen saatavuus kehittäjille ja käyttäjille. Puheentunnistus tekniikan alentunut hinta selittää osittain kiihtyvää kehitystä. 90-luvulla IBM Speech Server-ohjelmisto mahdollisti käyttöjärjestelmän ohjaamisen puheen avulla, mutta sen hinta oli 2500 dollaria käyttäjää kohden. (TIVI 2018.) Nykyisin kehittäjille on saatavilla merkittävästi tehokkaampia ja monipuolisempia ratkaisuja huomattavasti halvemmalla hinnalla tai jopa ilmaiseksi.

Puheentunnistuksen avulla voidaan helposti nopeuttaa yksinkertaisia toimenpiteitä kuten hälytysten asettamista puhelimeen tai säätietojen hakemista. Puheentunnistus voi helpottaa myös pitkien tekstien kirjottamista, koska sanelemalla on usein nopeampaa saada haluamansa sisältö aikaiseksi kuin kirjoittamalla.

Tässä työssä toteutettu sovellus tehtiin kokeellisessa mielessä ja tarkoituksena oli testata miten puheentunnistusta ja syntetisointia voidaan hyödyntää erilaisten tehtävien toteuttamiseen ja yksinkertaisten toimintojen aikaan saamiseksi. Sovelluksen avulla on mahdollista esimerkiksi sanella viesti ja kopioida viesti laitteen leikepöydälle puhutun komennon avulla. Lisäksi sovellus osaa lukea ääneen sille annettuja tekstejä sekä sen avulla on mahdollista muuttaa puheentunnistuksen asetuksia ääneen puhuttujen komentojen avulla.

Työn haastavin vaihe oli saada sovellus toteuttamaan ennalta määriteltyjä komentoja. Googlen puheentunnistusrajapinta tunnisti ainoastaan puheen, mutta se ei osannut eritellä, mitä puhe tarkoitti. Facebookin Wit.ai NLP alusta toimi taustajärjestelmänä, joka määritti viestin merkityksen, mutta mitä laajempi tuki erilaisille käskyille sovelluksessa on, sitä monimutkaisemmaksi komentojen tunnistaminen sovelluksen sisällä muuttui.

Tulevaisuudessa sovelluksen käyttöä on tarkoitus laajentaa toimimaan erillisen taustajärjestelmän kanssa, joka voi lähettää ja vastaanottaa erilaisia hälytyksiä työssä kehitetyn sovelluksen kautta. Nämä hälytykset voivat tulla esimerkiksi kiinteistössä sijaitsevista murto tai palohälyttimistä, jolloin käyttäjä voi kuunnella ja kuitata viestin sovelluksen kautta, ilman että käyttäjän täytyy kirjoittaa tai lukea pitkiä viestejä mahdollisissa kiiretilanteissa.

LÄHTEET

Able Here 2018. Speech recognition technology for disabled people [viitattu 21.3.2019]. Saatavissa: https://www.ablehere.com/latest-disability-news/660-speech-recognition-technology-for-disabled-people.html

Avenla Oy 2019. Avenla Oy [viitattu 8.3.2019]. Saatavissa: https://www.avenla.fi/avenla-oy/

Barr, L. 2019. 3 ways to stop smartphone use in the car [viitattu 23.3.2019]. Saatavissa: https://www.progressive.com/lifelanes/on-the-road/tame-smartphone-use-in-the-car/

Beal, V. 2019. TTS - text to speech [viitattu 16.2.2019]. Saatavissa: https://www.webopedia.com/TERM/T/TTS.html

Bidelman, E. 2019. Web apps that talk - Introduction to the Speech Synthesis API [viitattu 15.2.2019]. Saatavissa: https://developers.google.com/web/updates/2014/01/Web-apps-that-talk-Introduction-to-the-Speech-Synthesis-API

Boyd, C. 2018. The Past, Present, and Future of Speech Recognition Technology [viitattu 21.3.2019]. Saatavissa: https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf

Call Centre Helper 2017. The Top Five Uses of Speech Recognition Technology [viitattu 23.3.2019]. Saatavissa: https://www.callcentrehelper.com/the-top-five-uses-of-speech-recognition-technology-1536.htm

Cryer, A. 2018. Speech synthesis explained [viitattu 16.2.2019]. Saatavissa: https://everything.explained.today/Speech_synthesis/

Eugene, P. 2019. What Are the Most Common Speech Recognition Problems? [viitattu 23.3.2019]. Saatavissa: https://www.wisegeek.com/what-are-the-most-common-speech-recognition-problems.htm

Filipeguelber 2017. Creating a conversational chatbot using Wit.ai [viitattu 20.2.2019]. Saatavissa: https://labs.bawi.io/creating-a-conversational-chatbot-using-wit-ai-6eba3c625f4f

Google Cloud 2019. Cloud Speech-to-Text [viitattu 21.3.2019]. Saatavissa: https://cloud.google.com/speech-to-text/

Grabianowski, E. 2006. How Speech Recognition Works [viitattu 7.2.2019]. Saatavissa: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition.htm

Heires, K. 2017. The Risks of Voice Technology [viitattu 16.2.2019]. Saatavissa: http://www.rmmagazine.com/2017/10/02/the-risks-of-voice-technology/

Järvinen, P. 2017. Puheentunnistusta tarvitaan – tarve on ilmeinen [viitattu 10.2.2019]. Saatavissa: https://www.mikrobitti.fi/uutiset/puheentunnistusta-tarvitaan-tarve-on-ilmeinen/8484874d-1115-3a8a-bc9d-0f21e2993634

Kaplan, T. & Biggin M. 2012. Computational Methods in Cell Biology [viitattu 9.2.2019]. Saatavissa: https://www.sciencedirect.com/topics/medicine-and-dentistry/hidden-markov-model

Krishnan, A. 2018. IoT Based Home Automation System With Speech Recognition [viitattu 20.2.2019]. Saatavissa: https://www.instructables.com/id/Iot-Based-Home-Automation-System-With-Speech-Recog/

Lawton, G. 2018. Evaluate speech-to-text services from AWS, Microsoft and Google [viitattu 16.2.2019]. Saatavissa: https://searchcloudcomputing.techtarget.com/tip/Evaluate-speech-to-text-services-from-AWS-Microsoft-and-Google

Le, D. 2016. Why do we use Hidden Markov Models for speech recognition? [viitattu 23.3.2019]. Saatavissa: https://www.quora.com/Why-do-we-use-Hidden-Markov-Models-for-speech-recognition

Lillback, M. 2017. Miksi emme käyttäisi hyväksi avointa ekosysteemiä? [viitattu 16.2.2019]. Saatavissa: https://www.tivi.fi/Kumppaniblogit/symbio/miksi-emme-kayttaisi-hyvaksi-avointa-ekosysteemia-6640608

Linguaenglish 2015. 10 reasons why English is such an important language [viitattu 23.3.2019]. Saatavissa: https://www.lingualearnenglish.com/blog/tips-to-learn-english/10-reasons-english-important-language/

McClendon, B. 2016. Getting started with conversational bots using Wit.ai [viitattu 20.2.2019]. Saatavissa: https://knightlab.northwestern.edu/2016/05/23/getting-started-with-conversational-bots-using-wit-ai/

Microsoft Azeure 2019. What is Azure? [viitattu 16.2.2019]. Saatavissa: https://azure.microsoft.com/en-us/overview/what-is-azure/

Mozer, T. 2011. Growth in mobile and cloud-based speech recognition fueling embedded speech technologies [viitattu 23.3.2019]. Saatavissa: http://www.embedded-computing.com/embedded-computing-design/growth-in-mobile-and-cloud-based-speech-recognition-fueling-embedded-speech-technologies

Mozer, T. 2015. Deploying speech recognition locally versus the cloud [viitattu 11.2.2019]. Saatavissa: http://www.embedded-computing.com/embedded-computing-design/deploying-speech-recognition-locally-versus-the-cloud

Mäntylä, J. 2018. Tautiepidemiat havaitaan pian droneilla ja syöpäsolut tekoälyllä – "Suuri murros on käsillä" [viitattu 10.2.2019]. Saatavissa: https://yle.fi/uutiset/3-10390000

Niinimäki, J. 2015. Tekniikan päivien avajaispuhe: Teknologian kehittyminen on muovannut yhteiskunnat ja ihmisten arjen nykyiseen muotoonsa [viitattu 16.2.2019]. Saatavissa: https://www.oulu.fi/blogs/teknologian-kehittyminen-on-muovannut-yhteiskunnat-ja-ihmisten-arjen-nykyiseen-muotoonsa

Pinola, M. 2011. Speech Recognition Through the Decades: How We Ended Up With Siri [viitattu 9.2.2019]. Saatavissa: https://www.pcworld.com/article/243060/speech_recognition_through_the_decades_how_we_ended_up_with_siri.html

Rajala, A. 2019. Yle-raati auttaa kehittämään yhdenvertaista tarjontaa kaikille – “Yhdessä teemme Ylestä paremman”. [viitattu 10.2.2019]. Saatavissa: https://yle.fi/aihe/artikkeli/2019/01/11/yle-raati-auttaa-kehittamaan-yhdenvertaista-tarjontaa-kaikille-yhdessa-teemme

Reference 2019. What Are Some Advantages of Speech Recognition? [viitattu 21.3.2019]. Saatavissa: https://www.reference.com/technology/advantages-speech-recognition-100fadc6b980a624

Rodgers, K. 2016. Hidden Markov Models in Bioinformatics [viitattu 21.3.2019]

Saatavissa: http://slideplayer.com/slide/4757446/

Sivasankaran, S. 2017. Why Is Speech Recognition Technology So Difficult to Perfect? [viitattu 10.2.2019]. Saatavissa: https://www.huffingtonpost.com/quora/why-is-speech-recognition_b_13873978.html?guccounter=1

Sonix Inc 2019. A short history of speech recognition [viitattu 21.3.2019]. Saatavissa: https://sonix.ai/history-of-speech-recognition

Sulopuisto, O. 2016. Kuuleeko Cortana? Näin toimii Microsoftin puheentunnistus [viitattu 7.2.2019]. Saatavissa: https://www.tivi.fi/Kaikki_uutiset/kuuleeko-cortana-nain-toimii-microsoftin-puheentunnistus-6573569

Shires, G. & Wennborg, H. 2012. Web Speech API Specification [viitattu 15.2.2019]. Saatavissa: https://w3c.github.io/speech-api/speechapi.html#introduction

Shires, G. 2019. Voice Driven Web Apps: Introduction to the Web Speech API [viitattu 15.2.2019]. Saatavissa: https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API

Tieteen termipankki 2014. Markovin piilomalli [viitattu 23.3.2019]. Saatavissa: http://tieteentermipankki.fi/wiki/Nimitys:Markovin_piilomalli

TIVI 2018. Tivi 25 vuotta sitten – puheentunnistuksen ongelmat oli jo ratkaistu [viitattu 8.3.2019]. Saatavissa: https://www.tivi.fi/Kaikki_uutiset/tivi-25-vuotta-sitten-puheentunnistuksen-ongelmat-oli-jo-ratkaistu-6752879

Wit.ai 2019a. Natural Language for Developers [viitattu 20.2.2019]. Saatavissa: https://wit.ai/

Wit.ai 2019b. Integrate Wit everywhere [viitattu 20.2.2019]. Saatavissa: https://wit.ai/docs/recipes#integrate-into-my-app

Wit.ai 2019c. </> HTTP API Reference [viitattu 22.2.2019]. Saatavissa: https://wit.ai/docs/http/20170307

Wit.ai 2019d. Built-in Entities [viitattu 22.2.2019]. Saatavissa: https://wit.ai/docs/built-in-entities/20180601

W3Schools 2019. JavaScript Switch Statement [viitattu 8.3.2019]. Saatavissa: https://www.w3schools.com/js/js_switch.asp

YLE. 2019. Tasa-arvo ja yhdenvertaisuussuunnitelma [viitattu 10.2.2019] Saatavissa: https://yle.fi/aihe/artikkeli/2018/01/05/tasa-arvo-ja-yhdenvertaisuussuunnitelma-2018-2019

Zohaib, R. 2016. Use Wit.ai Natural Language Processing in Elixir for bots [viitattu 20.2.2019]. Saatavissa: https://zohaib.me/use-wit-ai-natural-language-processing-in-elixir-for-bots/