CHALMERS

Virtuell assistent för e-handel

Examensarbete inom Data- och Informationsteknik

FILIP TÖRNQVIST
TOM BJURENLUND

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2019

EXAMENSARBETE

Virtuell assistent för e-handel

FILIP TÖRNQVIST
TOM BJURENLIND

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2019

Virtuell assistent för e-handel
FILIP TÖRNQVIST
TOM BJURENLIND

Examinator: Jonas Duregård

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Institutionen för Data- och Informationsteknik
Göteborg 2019

SAMMANFATTNING

Den här rapporten beskriver implementeringen av en virtuell assistent med avsedd användning som kundsupport inom e-handel. Syftet med projektet var att applicera nya tekniker, såsom AI, i implementeringen av en virtuell assistent. Arbetet med projektet innefattade bland annat urval, instudering och användning av en rad olika verktyg, ramverk, program och tjänster. I projektet användes en AI-baserad tjänst, som med hjälp av artificiella neurala nätverk och maskininlärning tillhandahåller språkförståelse för den virtuella assistenten som är resultatet av detta projekt.

Nyckelord: e-handel, maskininlärning, artificiella neurala nätverk, AI, virtuell assistent

ABSTRACT

This report describes the implementation of a virtual assistant with intended use as an e-commerce customer support. The purpose of the project was to apply new technologies, such as AI, in the implementation of a virtual assistant. The work on the project included selection, study and use of a variety of tools, frameworks, programs and services. In the project, an AI-based service is used, which, using artificial neural networks and machine learning, provides language understanding for the virtual assistant that is the result of this project.

Keywords: e-commerce, machine learning, artificial neural networks, AI, virtual assistant

FÖRORD

Detta projekt utförs som ett examensarbete i det treåriga datateknik programmet på Chalmers tekniska högskola i Göteborg, Sverige. Datateknik programmet är en grundutbildning och omfattar matematik, ellära, datorteknik och programmering. Examensarbetet sker i samarbete med företaget 3bits Consulting AB och omfattar utveckling och implementering av en virtuell assistent för kundservice, vars syfte är att hjälpa personer att hitta produkter de är intresserade av. Ett stort tack till företaget 3bits för utlåning av sina lokaler och utrustning, samt ett stort tack till Sofia Winterlén (3bits marknadskommunikation), Erik Jorpes (3bits handledare), Joachim von Hacht (chalmers handledare) och alla andra som har hjälpt oss under projektets gång.

Innehållsförteckning

Beteckningar

Teoretiska beteckningar

Artificiell intelligens (AI) - Imitering av naturlig intelligens.
Neuronnät - Artificiella neuronnät, självlärande datoralgoritm.
Bot - Virtuell robot, datorprogram som autonomt utför uppgifter.
E-handel - Elektronisk handel, handel via datornätverk.
Virtuell assistent (VA) - IT-tjänst som är utformat för att föra en hjälpfull konversation med en användare, även känd som chattbot.
Molntjänster - IT-tjänster som är tillgängliga för användaren på begäran via internet.
Webbapplikation - Applikation som är ämnad för kommunikation över internet.
Natural Language Understanding (NLU) - Artificiell intelligens som utför tolkning av ett naturligt språk.

Tekniska beteckningar

Structured Query Language (SQL) - Ett programmeringsspråk för hantering av data i en relationsdatabas.
Non Structured Query Language (NoSQL) - Till skillnad från SQL är NoSQL inte ett programmeringsspråk, det är en databas som inte baserar datastrukturen på relationer.
Dokumentdatabas (DDB) - En variant av en NoSQL databas, dvs. en databas utan relationer mellan data.
C# - C-sharp, ett objektorienterat programmeringsspråk utvecklat av Microsoft.
Visual Studio - Microsoft Visual Studio 2017, integrerad utvecklingsmiljö för utvecklad av Microsoft.
Azure - Microsoft Azure, molntjänster från Microsoft.
LUIS - Language Understanding Intelligent Service, natural-language understanding artificiell intelligens byggd av Microsoft.
Javascript Object Notation (JSON) - En datalagringsstruktur i text.
Newtonsoft - Ett mjukvarubibliotek som utför tolkning av JSON formatet för C#.
NuGet - Ett hanteringsprogram för mjukvarubibliotek.
HTTP - Hypertext Transfer Protocol, ett kommunikationsprotokoll för klienter och servrar.
HTTP-GET - GET-metod, en HTTP-metod som är avsedd för att hämta information från en server.

Figurer

Tabeller

1 Inledning

Fler och fler företag använder sig av konversationella, virtuella assistenter (VA). Telia, PostNord och SEB använder sig av VA systemet Humany [1] för att automatisera specifika processer inom företaget, såsom kundsupport, kundfeedback, produktsökning och produktköp. Fördelarna med att använda en virtuell assistent för olika företagstjänster är bland annat lägre lönekostnader och tillgodoseende av viktiga företagstjänster som kundsupport, sökning i produktutbud, beställning av produkter samt kundfeedback under hela dygnet. En VA kan dessutom ge ett bättre gränssnitt för att hjälpa en kund hitta det de söker efter genom en bekant språk- eller text-dialog. En annan anledning för företag att introducera en VA är att dessa enkelt kan integreras i existerande meddelandetjänster, såsom Facebook Messenger. Detta exponerar företagets tjänster till en plattform där en stor del av kunderna redan finns [2].

Artificiell intelligens (AI) är ett samlingsnamn för olika algoritmer och tekniker som härmar mänsklig intelligens. AI kan användas till att lösa problem mer som människor. Detta kan åstadkommas på ett flertal sätt. Antingen genom att hårdkoda sannolika beteenden, eller genom att dynamiskt lära programmet med hjälp av maskininlärning. Maskininlärning går ut på att ett artificiellt neuronnät tränas på en stor mängd data, och på så sätt ges förmågan att kunna lösa olika komplexa problem. Ett exempel på ett vanligt sådant problem är konvertering av röst till text och från text till röst [3]. Ett exempel på ett löst definierat problem där användning av AI skulle kunna förbättra utfallet är personliga kundrekommendationer.

Med användning av AI går det att åstadkomma personifierade produktrekommendationer, baserat på data från en användarprofil; med information om till exempel kön, ålder, stad, preferenser, tidigare sökningar och tidigare köp. Detta ger företag inom e-handel möjligheten att kunna prestera bättre, då mer relevanta produktrekommendationer ökar chansen till att sälja produkter till kunden [4].

Företaget 3bits Consulting AB är ett teknikföretag som specialiserar sig på e-handel, och har funnits i över tio år. Under denna tiden har de vuxit från tre till fyrtio anställda. Inom deras arbetsområde finns mest utveckling av e-handelssystem för många stora kunder såsom Lindex och Tingstad [5]. De har annonserat en uppgift angående utveckling av en virtuell assistent för e-handel vilket är grunden till detta projekt.

1.1 Syfte

Projektets syfte är att undersöka hur nya molnbaserade AI-tjänster för exempelvis språkförståelse kan användas för att skapa dialoger inom e-handeln.

1.2 Mål

Målet med projektet är att utveckla en virtuell assistent för internetbaserade meddelandetjänster avsedd för användning inom e-handel. Den virtuella assistenten skall kunna ge personliga produktrekommendationer till användaren i någon form. Den virtuella assistenten skall använda sig utav AI-tekniker för språkförståelse för att kunna tillhandahålla de personliga rekommendationerna. Den VA:en skall dessutom kunna urskilja och besvara vanliga fraser och dialoger, exempelvis hälsningsfraser, uppdatering av profil och avbrytande av nuvarande dialog.

1.3 Avgränsningar

Den virtuella assistenten skall vara kapabel till att ge personliga produktrekommendationer till användaren, samt stödja fundamentala konversationella dialoger; såsom hälsning, bekräftande- och avbrytande- av nuvarande dialog. Den virtuella assistenten kommer dessutom inledningsvis ställa personliga frågor och spara denna information för senare användning i en databas. VA:n kommer dessutom samla in och spara information från sökningar och andra handlingar kunden gör. Denna data skall sedan användas av den virtuella assistenten för att tillgodose de personliga rekommendationerna. Projektet kommer inte att behandla implementering av ett artificiellt neuralt nätverk på egen hand, utan färdiga molntjänster för detta kommer att användas. Implementationen kommer dessutom inte att jämföras med en mänsklig kundsupport.

2 Metod

I inledningsfasen skall hela projektet planeras och avgränsas i samarbete med handledare på 3bits. Därefter skall en tidsplan tas fram. I tidsplanen beskrivs hur arbetet med projektet skall struktureras: var, när och i vilken ordning arbetet skall utföras. En viktig del i arbetet blir att komma fram till vilka API:er, ramverk, tjänster, tekniker och verktyg som skall användas för att genomföra projektet.

När detta är avklarat skall design och implementation av den virtuella assistenten påbörjas. Under implementationen skall en testdriven metodik användas. Under implementationen skall nödvändiga tekniker, koncept och tjänster instuderas vidare efter behov. Genomgående testning av samtliga tillägg av ny funktionalitet skall utföras innan kod accepteras.

Målet med projektet anses ha mötts om en fungerande virtuell assistent har tagits fram som kan tillhandahålla kundservice inom e-handel för kunder med hjälp av AI-tekniker för språkförståelse. Den virtuella assistenten skall specifikt kunna tillhandahålla sökningar av olika produkter och kunna urskilja olika vanliga fraser från användaren, exempelvis hälsningsfraser, uppdatering av profil och avbrytande av nuvarande dialog.

Målen kommer att nås genom att träna språkförståelse-komponenten av VA:en att urskilja de avsikter som bedöms som mest användbara hos användaren. Denna träning kommer specifikt att åstadkommas genom att specificera de avsikter som VA:en bedöms skall kunna behandla, och sedan ange exempelmeningar till varje avsikt. När tillräckligt med exempelmeningar har angetts till varje avsikt kommer språkförståelse-komponenten att tränas genom maskininlärning att koppla ihop liknande meningar till tillhörande avsikt. Effektiviteten i denna process kommer att bero på mängden exempel som tränas på och hur generella dessa exempel är. i avsnittet Teoretisk bakgrund nedan beskrivs denna process mer omfattande.

3 Teoretisk bakgrund

I denna del presenteras teoretiska begrepp som används inom området.

3.1 Artificiella neurala nätverk

Artificiella neurala nätverk är en teknik som strävar till att efterlikna hur en biologisk hjärna fungerar. Det är med andra ord ett artificiellt nätverk av neuroner, där varje neuron kan utföra en enkel beräkning. Dessa neuroner kan vara organiserade på ett oändligt antal sätt, och skilja till antal, men en vanligt variant är att de är organiserade i lager, där varje lager av neuroner passar vidare sina beräkningar till nästa lager. Vikten av anslutningarna mellan olika neuroner är vad som huvudsakligen tränas vid så kallad *maskininlärning*, vilket beskrivs kortfattat nedan.

3.2 Maskininlärning

Maskininlärning innebär att ett artificiellt neuralt nätverk tränas till att utföra någon specifik uppgift. Detta åstadkoms genom att ett artificiellt neuralt nätverk matas med en stor mängd *input* och korresponderande *output* information. Under denna process uppdateras nätverkets parametrar på ett lämpligt sätt för att minimera felet i nätverkets *output*. Uppdateringen av varje parameter beräknas genom derivering av felet med avseende på motsvarande parameter. Felet definieras som skillnaden mellan det faktiska värdet nätverket kom fram till för en viss *input* av data, och det önskade, fördefinierade värdet för denna data. De huvudsakliga parametrar som uppdateras är vikterna av anslutningarna mellan olika neuroner. Resultatet av detta är att det artificiella neurala nätverket lär sig att ge en viss *output* för liknande, men inte nödvändigtvis samma, *input*. Det går också att se det som att nätverket lär sig mönster i datan relevant för uppgiften, vilket hjälper det att generalisera till annan, liknande data.

3.3 Artificiell intelligens

Artificiell intelligens (AI) är en benämning för ett antal algoritmer som baseras på att försöka efterlikna mänsklig intelligens. Detta åstadkoms ofta idag genom att ett artificiellt neuralt nätverk tränas genom maskininlärning till att utföra särskilda, komplexa uppgifter. Exempel på användningsområden är språkförståelse, kategorisering och förståelse av stora mängder högdimensionell data, samt identifiering och kategorisering av objekt i bilder.

3.3.1 Förståelse av naturliga språk

Språkförståelse m.h.a. Natural language understanding (NLU) AI syftar till att härleda vad en användare har för avsikt med ett eller flera meddelanden, samt extrahera viktiga data från dessa dialoger, såsom namn och adresser. Detta sker i syfte att exempelvis tillhandahålla smarta, virtuella assistenter, samt för att effektivt översätta text från ett språk till ett annat. Detta kan uppnås genom att ett artificiellt neuralt nätverk tränas genom maskininlärning till att koppla ihop exempel på fraser till tillhörande avsikt. Tanken är sedan att nätverket har generaliserats till att kunna koppla ihop arbiträra meningar till tillhörande avsikt.

3.3.2 Produktrekommendationer

Produktrekommendationer syftar till att identifiera vad en kund mest troligt är intresserad av för typ av produkter. Rekommendationerna kan trivialt skapas genom att prioritera produkter som har attribut som hittas i tidigare köp eller gillade produkter, samt en

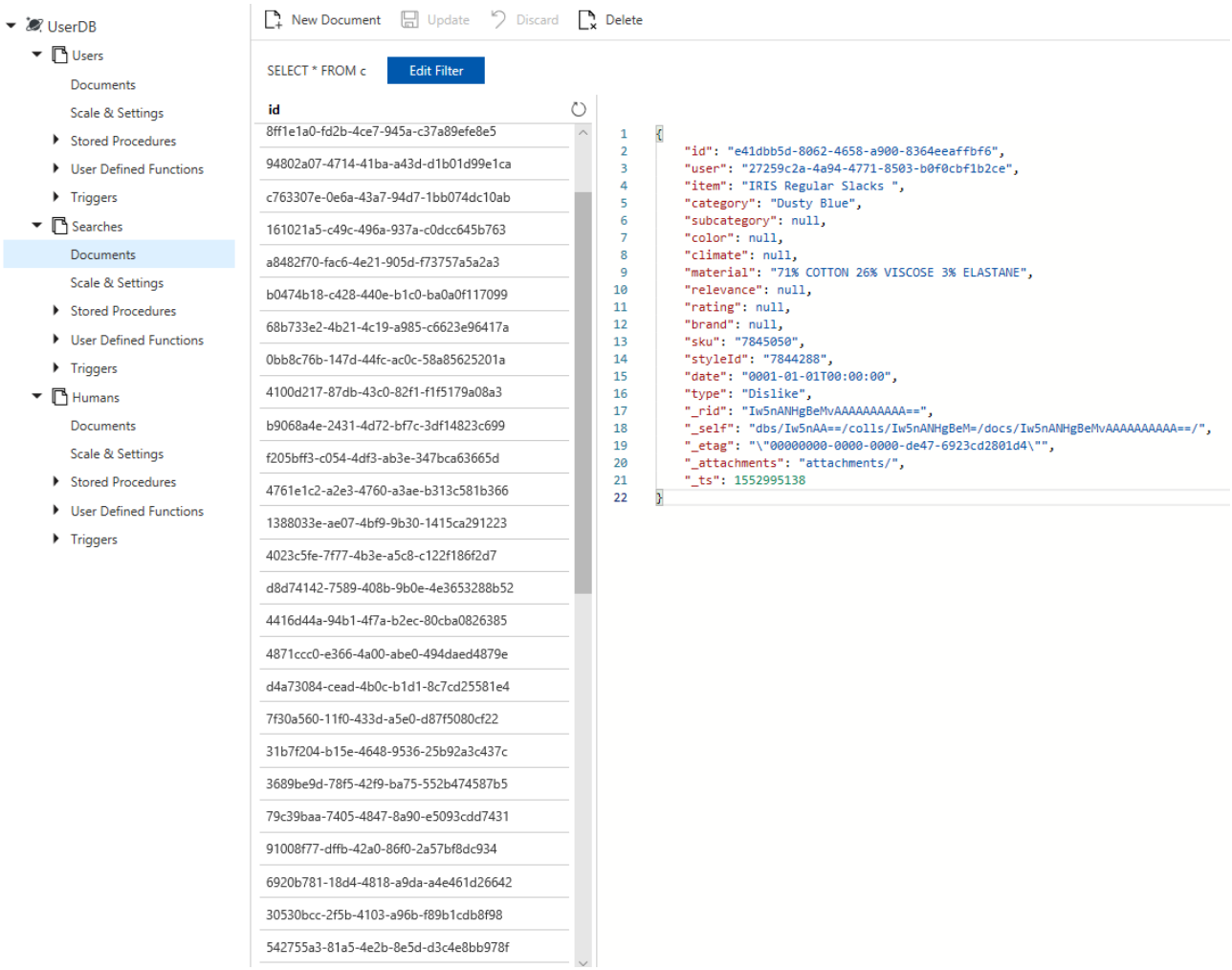
pålagd prioritet ökning för produkter som matchar användarens kön, ålder och preferenser. En alternativ metod för att hitta produktrekommendationer är att låta ett artificiellt neuralt nätverk tränas genom maskininlärning till att prioritera produkter som kunden visat intresse för tidigare eller som passar in på användarens profil. Samtidigt tränas nätverket till att minska prioriteten på produkter som kunden direkt eller kundens profil indirekt indikerat är ointressanta. Förhoppningen av detta är att nätverket skall generaliseras till att kunna avgöra om en viss produkt liknar någon av de produkter kunden tidigare har visat intresse för, och därmed är av intresse. Att tillämpa AI-tekniker likt vad som beskrevs ovan för att ta fram de personliga rekommendationerna var en alternativ lösning som beaktades att implementeras och utvärderas under projektet, men som kastades åt sidan i fördel för den triviala lösning.

4 Teknisk bakgrund

Flera olika tjänster var möjliga att använda i projektet för att implementera den virtuella assistenten. Bland annat behövdes ett ramverk för att utveckla den virtuella assistenten, en plattform för hosting av den virtuella assistenten och en tjänst för språkförståelse för att härleda vad en användare vill göra. Det behövdes dessutom en meddelandetjänst med tillhörande integrations-API för att kunna kommunicera med assistenten. Nedan följer en kortfattad översikt över några relevanta tekniker.

4.1 Dokumentdatabaser

En dokumentdatabas (DDB) tillåter ett mer flexibelt sätt att arbeta med data. En traditionell relationsdatabas kräver att utvecklaren designar och utformar ett schema för databasen (tabeller, relationer mellan tabeller, datatyper, indexes, nycklar, etc.). En DDB tillåter istället utvecklaren att fritt och flexibelt spara data direkt i databasen, i så kallade kollektioner, i valfritt format. En DDB presenterar heller inte några krav på att dataformatet av varje dokument i en viss kollektion (motsvarande tabell i en relationsdatabas) skall vara samma. Det är alltså möjligt att ha flera olika dokument i samma kollektion med skilda format, något som inte är möjligt i en traditionell SQL-databas. En DDB sparar datan i ett JSON-liknande format. Flexibiliteten med att kunna spara nästan vad som helst, i vilket format som helst, kan dock presentera problem med konsistens, något den striktare SQL-databasen är mycket bättre på att undvika [6]. Sökningar går till genom att utvecklaren specificerar vilken kollektion denne vill söka i, och sedan vilka värden olika attribut för de önskvärda dokumenten skall ha. Exempel på hur en DDB kan se ut visas nedan i figur 1.



Figur 1: Utdrag ur en dokumentdatabas. *Users*, *Searches* och *Humans* är kollektioner och ett dokument ur kollektionen *Searches* visas i figuren längst till höger.

4.2 Microsoft Azure

Microsoft Azure är en plattform med Microsofts molntjänster som innehåller verktyg, ramverk och API:er för utveckling av bl.a. webbapplikationer, dessa tjänsterna kallas resurser i plattformen och kostar pengar. Azure tillhandahåller även hosting och automatisk underhåll av driftsatta webbapplikationer [7]. Azure går att nå med ett kostnadsfritt utvecklarkonto vilket innehåller 1750 kr i fri kredit samt 25 dygn av begränsad användning av ett flertal tjänster kostnadsfritt, alternativt kan ett *Pay-As-You-Go*-konto användas där alla tjänster är tillgängliga men kostar pengar över en viss kvota.

4.2.1 Microsoft Bot Framework

Microsoft Bot Framework är ett ramverk från Microsoft Azure som innehåller funktionalitet för att kunna skapa bl.a. virtuella assistenter [8]. Ramverket kan konfigureras till att koppla upp sig mot olika meddelandetjänster, exempelvis Telegram Messenger. Web App Bot är en resurs i Microsoft Azure och fungerar som en bas-implementering av Microsoft Bot Framework. Den används som en startmall till en VA och den innehåller en primitiv dialog funktion men är inte uppkopplad till några andra tjänster.

4.2.2 Azure Cosmos DB

Azure Cosmos DB är en globalt distribuerad databastjänst i Microsoft Azure, och tillhandahåller API:er för olika databaser, t.ex. SQL, MongoDB, Cassandra, Tables samt Gremlin. Den underliggande databasmodellen är en dokumentdatabas, men går att kommunicera med genom någon av ovanstående API:er genom ett tillhandahållet bibliotek. Det går alltså att exempelvis konstruera SQL-förfrågningar i koden, vilka biblioteket sedan omvandlar till ett generellt format för att kunna köras mot den underliggande dokumentdatabasen [9].

4.2.3 Azure Cosmos DB Emulator

Azure Cosmos DB Emulator är en emulator som tillåter utvecklaren att arbeta med Azure Cosmos DB lokalt, och därmed inte behöva betala för tjänsten i Azure under utvecklingsfasen [10]. Emulatorn körs som en tjänst i bakgrunden, där den bland annat lyssnar på förfrågningar. Emulatorn tillhandahåller ett webbaserat användargränssnitt för att visualisera eventuella databasstrukturer och data som kan finnas i databasen.

4.5.4 Azure Bot Framework Emulator

Azure Bot Framework Emulator är en emulator som tillåter utvecklaren att kommunicera med den virtuella assistenten [11]. Emulatorn körs som en vanlig applikation, och är alltså inte en bakgrundstjänst som Azure Cosmos DB emulatorn. Emulatorn tillhandahåller användargränssnitt för att skicka textmeddelanden till den virtuella assistenten.

4.2.5 LUIS

LUIS är en tjänst från Microsoft Azure som tillåter utvecklaren att skapa, specificera, träna och kommunicera med AI-modeller för språkförståelse [12]. Tjänsten tillåter utvecklaren att specificera vilka avsikter som tjänsten skall kunna identifiera från textmeddelanden som skickas till tjänsten. Microsoft tillåter 10 000 API-anrop per månad gratis och överstigande av denna kvota kostar pengar.

Exempel på fraser anges till varje specificerad avsikt, och tjänsten tränas därefter på att koppla ihop avsikterna med fraserna genom maskininlärning. Tjänsten kan sedan identifiera avsikt och annan viktig information, exempelvis identifierade kategorier, adresser, priser och produktnamn från meddelanden som skickas till denne. Avsikt och information skickas sedan tillbaka till de webbapplikationer som skickar textmeddelanden till tjänsten. Om exempelvis meddelandet “Show me some jeans” skickas till tjänsten, skall den identifierade avsikten vara *BrowseProducts*, och “jeans” skall vara en identifierad entitet.

4.3 Telegram Messenger

Meddelandetjänsten, som förutom att tillåta kommunikation mellan två eller flera parter, tillhandahåller ett API för integration av virtuella assistenter [13].

4.3.1 Telegram Messenger Bot API

API:et tillåter utvecklare att integrera sina virtuella assistenter med Telegram Messenger. Därefter kan användare kommunicera med dessa direkt genom Telegram Messenger [14]. Kommunikationen är JSON-baserad, och meddelanden tas emot i webbapplikationen som JSON-objekt. En VA skapas genom att utvecklaren, från ett vanligt Telegram-konto, skriver in kommandon i form av textmeddelanden till den så

kallade *BotFather*. *BotFather* är en VA som kan kommuniceras med genom Telegram Messengers chattprogram. Exempelvis skapas en VA genom kommandot *newbot*.

5 Kravspecifikation

Uppdragsgivaren föredrog att Microsofts produkter skulle användas och att programmeringsspråket C# skulle appliceras för att implementera den virtuella assistenten eftersom dessa produkter används redan i företaget. Uppdragsgivaren förväntade sig att den virtuella assistenten skulle vara kapabel att tillhandahålla sökning och visning av produkter genom AI-tekniker för språkförståelse. VA:en förväntas även kunna känna igen och besvara vanliga fraser och dialoger, exempelvis hälsningsfraser, uppdatering av profil och avbrytande av nuvarande dialog. Ytterligare ett krav var att projektet utfördes på uppdragsgivarens arbetsplats och att uppdragsgivarens lokala databas för produkter utnyttjas på ett sparsamt och hållbart sätt. Förutom ovan nämnda krav och rekommendationer från uppdragsgivaren hade projektet fria tyglar och egna idéer uppmuntrades.

6 Genomförande

I den initiala fasen av projektet diskuterades och planerades avgränsningar för den virtuella assistenten. Tillsammans med handledare på företaget, skapades en tidsplan för projektet samt planerades hur arbetet med projektet skulle struktureras samt var och när arbetet skulle utföras.

Därefter fortsatte projektet med att utvärdera olika tjänster, API:er, systemarkitekturer och ramverk relevanta för projektet. Detta följdes av val mellan de olika utvärderade alternativen. Parallellt med detta skapades privat lagringsutrymme på GitHub och Azure konton.

Efter instudering av API:erna och ramverken fortsatte projektet med att börja dela upp och arbeta med de initiala arbetsuppgifterna som hade formulerats i tidsplanen. Arbetet med arbetsuppgifterna fortsatte sedan enligt tidsplaneringen, parallellt med inlärning av diverse nya koncept vid behov.

6.1 Val av tjänster, API:er och ramverk

Nedan följer en beskrivning av varför de tjänster, API:er och ramverk som användes i projektet valdes.

6.1.1 Ramverk för virtuell assistent

De ramverk som togs i beaktande var Googles *DialogFlow,* IBM *Watson Assistant* och *Microsoft Bot Framework*. I slutändan valdes Microsoft Bot Framework då det hade stöd för programmeringsspråket C# och för att det var från Microsoft.

6.1.2 Tjänster för språkförståelse

Tjänsterna Microsoft LUIS och IBM Watson utvärderades, och slutligen valdes Microsoft LUIS då det gav bättre kompatibilitet med Visual studio och C#. LUIS var dessutom fri att användas så länge 10 000 anrop till tjänsten ej överskreds varje månad.

6.1.3 Plattform för hosting av virtuell assistent

Microsoft Azure valdes ut som plattform för hosting av den virtuella assistenten eftersom Microsoft LUIS är specialbyggd för att användas av denna plattform. Microsoft Azure är även väldigt flexibelt när det kommer till att testa molntjänster kostnadsfritt under utvecklingsfasen. Då Microsoft är leverantör av samtliga utvalda tjänster var det enkelt att konfigurera alla tjänster till att samverka.

6.1.4 API för integration med meddelandetjänst

API:et för integration mellan meddelandetjänsten som valdes är helt inbyggt i Microsoft Bot Framework. API:et är dessutom fullt användbart efter minimal konfiguration.

6.1.5 Meddelandetjänst

Det beslutades att använda Telegram Messenger som meddelandetjänst i projektet. Anledningen till att Telegram valdes var för att det var den meddelandetjänst som krävde minst tid att använda. De andra meddelandetjänsterna som beaktades krävde noterbart mer konfiguration. Facebook Messenger krävde dessutom ett kontrakt för att överhuvudtaget kunna använda tjänsten. Detta sker genom att utvecklaren anger

information om sin applikation samt hur applikationen kommer använda användarinformation, detta skickas in till företaget bakom meddelandetjänsten för utvärdering och eventuellt skapande av kontrakt.

6.2 Utvecklingsmiljö

På anmodan av företaget valdes Visual Studio då detta är den utvecklingsmiljö de använder sig av. Visual Studio stödjer programmeringsspråket C# och är en väletablerad utvecklingsmiljö som har stöd för flertalet tjänster vilka underlättar utvecklingen, exempelvis dess *debugger*. Visual Studio är även den framtagen av Microsoft och har därmed har den inbyggt stöd för att arbeta direkt med Microsoft Azure, exempelvis enkel distribuering av webbapplikationen direkt till molnet.

Under utvecklingsprocessen användes två emulatorer för att möjliggöra lokal, kostnadsfri utveckling av den virtuella assistenten. Den ena emulatorn emulerade användardatabasen, medan den andra emulatorn möjliggjorde kommunikation med webbapplikationen. Kostnaderna för att köra användardatabasen och webbapplikationen i Azure under hela utvecklingsfasen hade varit hög, så dessa två emulatorer var viktiga för projektets genomförande. För utveckling under projektet användes två maskiner med Windows 10.

6.2.1 Visual Studio

Konfigureringen av utvecklingsmiljön påbörjades genom att registrera utvecklingskonton till Microsoft Visual Studio och därefter installerades Microsoft Visual Studio 2017 med standardinställningar på maskinerna. Mjukvarubiblioteket Newtonsoft laddades ner till Visual Studio genom tilläggshanteringsprogrammet NuGet. Newtonsoft är ett mjukvarubibliotek som läser och skriver JSON. Biblioteket användes i projektet för att omvandla C#-objekt till JSON-objekt, och från JSON-objekt till C#-objekt.

6.2.2 Driftsättning av Azure

Ett Microsoft Azure utvecklarkonto registrerade, därefter skapades en resursgrupp för att hålla projektets delar. Resursgruppen delades sedan med alla medlemmar för att tillåta ett flertal personer att koppla upp sig mot projektets resurser. Det kostnadsfria utvecklarkontot uppgraderades till ett *Pay-As-You-Go*-konto efter de 25 gratis dygnen hade gått ut.

Driftsättning av Bot Framework skedde genom att Azure-resursen Web App Bot skapades i webbportalen för Microsoft Azure och laddades ner till Visual Studio. Därefter var en primitiv version av en VA redo att användas och utvecklas vidare.

En resurs för Azure Cosmos DB skapades först webbportalen för Microsoft Azure, precis som *Web App Bot* resursen, och en mall för att direkt kunna arbeta med databasen var tillgängligt att laddas ned. Mallen importerades in i det existerande Visual Studio projektet. I projektets konfigurationsfil behövde en ändpunkt och nycklar för den skapade resursen anges. Då tjänsten kostade pengar raderades resursen och lokal utveckling m.h.a. Azure Cosmos DB Emulator användes istället.

LUIS-tjänsten skapades sedan genom webbportalen för LUIS. Vilket därefter konfigurerades initialt med standard intents och entities, när detta var gjort klickades publish för att driftsätta instansen av tjänsten. Den ovannämnda *Web App Bot*-resursen behövde sedan konfigureras med ändpunkt och nycklar till den skapade LUIS-instansen

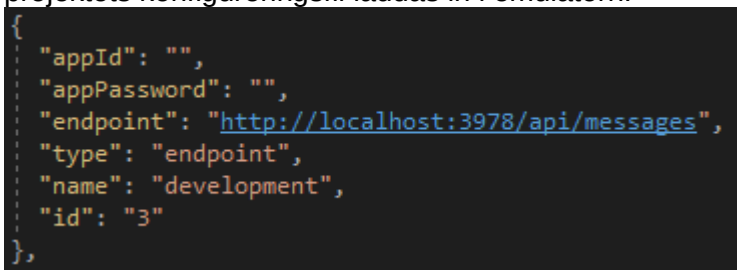
för att tillåta kommunikation. Demo-projektet går nu att utveckla vidare med funktionalitet, exempelvis olika dialoger.

6.2.3 Användning av emulatorer

Nedan följer en beskrivning av hur de två emulatorerna användes.

6.2.3.1 Bot Framework Emulator (V4)

Den emulator som användes för att emulera ett frontend var Bot Framework Emulator (V4). Konfigurering av emulatorn skedde genom att projektets konfigureringsfil (.bot-fil) ändrades genom att vid *development*-läget signalerar konfigurationen att emulatorn skall användas som endpoint, se figur 2. Därefter skall emulatorn konfigureras genom att projektets konfigureringsfil laddas in i emulatorn.



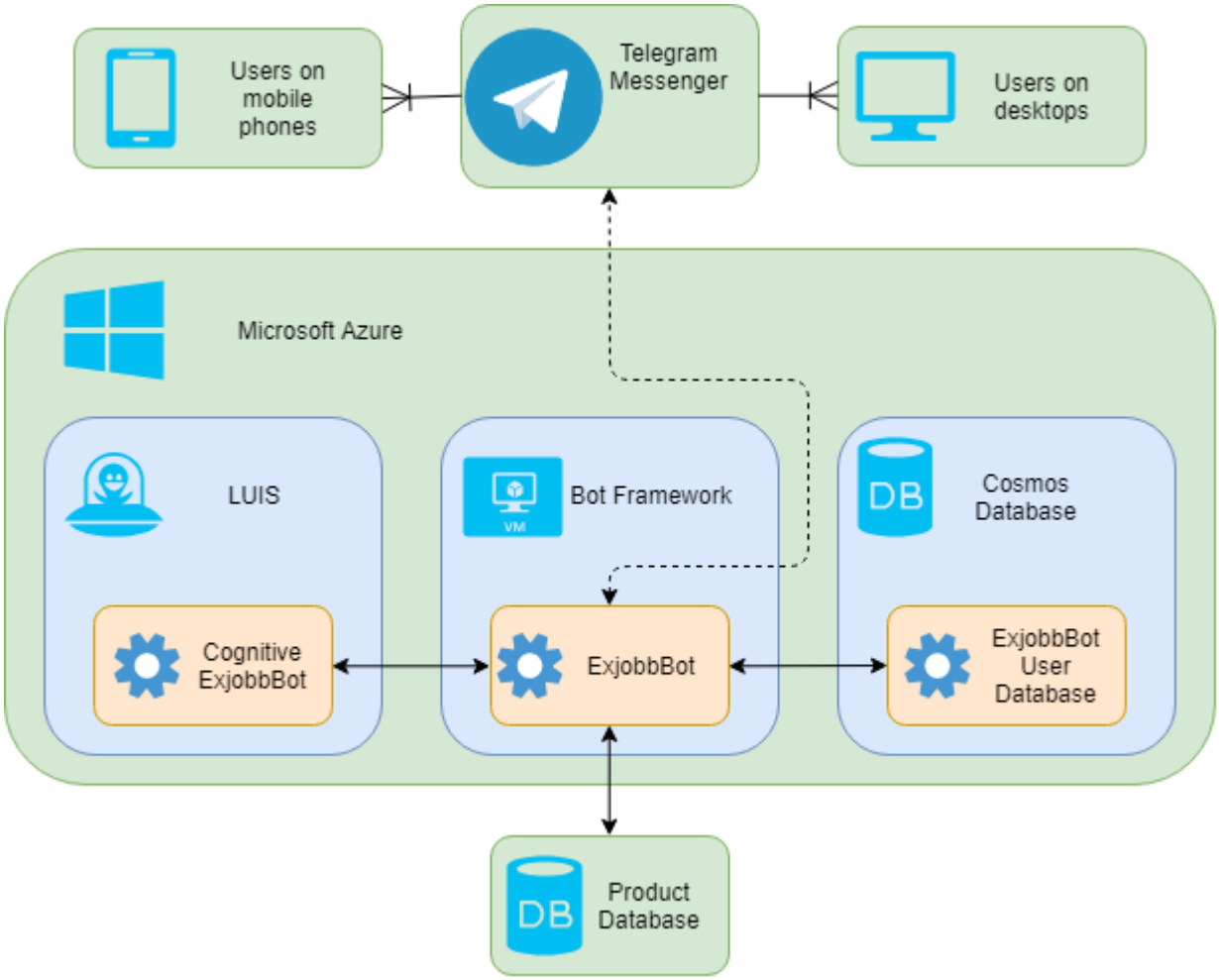
Figur 2: Konfigurering av emulator.

6.2.3.2 Azure Cosmos DB Emulator

Med användning av Azure Cosmos DB Emulator, tilläts det att arbeta med databasen lokalt, vilket jämfört med molndatabasen var både snabbare och kostnadsfritt. Emulatorn krävde enbart att projektet konfigurerades att använda loopback-adressen localhost med porten 8081, istället för att koppla upp till Microsoft Azure.

6.3 Systemarkitektur

Den virtuella assistenten är uppbyggd av tre delar varav en är en webbapplikationen, *ExjobbBot* i figur 3. Denna samverkar med en användardatabas, *ExjobbBot User Database* i figur 3, och en LUIS del, *Cognitive ExjobbBot* i figur 3. Dessa delkomponenter är hostade i Microsoft Azure och har därav tillgång till varandra och andra Azure tjänster, till exempel som det uppkopplings-API som används för att koppla ihop en Microsoft Bot Framework applikation och Telegram Messenger.



Figur 3: Diagram på systemarkitektur, där orange utgör projektet.

6.3.1 ExjobbBot

ExjobbBot körs i Azure-molnet, och tar emot alla meddelanden från användare. ExjobbBot använder sig av Cognitive ExjobbBot, ExjobbBot User Database och Product Database för att konstruera en respons tillbaka till användaren. kommunikationen med LUIS och databaserna sker genom direkt med funktionsanrop utan mellanhänder, ifyllda pilar i figur 3.

6.3.2 ExjobbBot User Database

I databasen sparas all användardata, såsom personliga uppgifter och aktivitet. Lagringen av informationen görs av ExjobbBot. Databasen körs under produktion i Azure.

Informationen samlas in genom hela konversationen och sparas temporärt i servern som exekverar koden. Det som samlas in är användarens namn, land, storlek på kläder, om användaren är intresserad av kläder till ett barn eller till en kvinna, samt alla handlingar användaren utför.

6.3.3 Cognitive ExjobbBot

Cognitive ExjobbBot är en instans av LUIS med inlärda exempel av tolkningar av meddelanden till motsvarande avsikter. Instansen körs även den i Azure och ingen lokal emulator tillhandahålls för tjänsten. Instansen används av ExjobbBot varje gång denne tar emot ett meddelande från en användare i syfte att tolka meddelandet.

6.3.4 Product Database

Product Database är en extern produktdatabas, denna innehåller all information om de produkter den virtuella assistenten kan visa. En del inom *ExjobbBot* är dedikerad till att kommunicera med produktdatabasen och enbart denna del sköter kommunikation med produktdatabasen. Databasen körs på servrar ägde av företaget som gav ut annonsen till examensarbetet. Därför krävdes en offentlig API-nyckel för att kunna kommunicera med databasen i produktion, men inte under lokal utveckling.

6.3.5 Uppkoppling till Telegram Messenger

Microsoft Bot Framework konverterar all kommunikation med externa chattprogram till ett standardiserat format innan det passas vidare till *ExjobbBot*, se den streckade pilen mellan *ExjobbBot* och *Telegram Messenger*.

6.3.6 Felhantering

Assistent-kodens felhanteringen sker genom att projektet inte kastar exceptions om det är möjligt att återhämta sig från problemet, utan istället konverteras allt till godtagbara *return* värden så som *null* eller tomma datastrukturer och felet rapporteras via funktioner i en statisk debug-klass, se figur 4, på så sätt fortsätter programmet fungera även när fel uppstår. Felhanteringen sker på detta sättet för att förtydliga vad som är fel i programkoden och vad som är fel på grund av externa faktorer, där programfel rapporteras genom funktionen *Bug* och externa fel rapporteras genom funktionen *Warning*.

```
namespace Exjobb
{
    public static class Dbg
    {
        public static void Message(string format, params object[] args)
        {
            MessageConsole(ConsoleColor.Yellow, "<MESSAGE> ", format, args);
        }

        public static void Warning(string format, params object[] args)
        {
            MessageConsole(ConsoleColor.Magenta, "<WARNING> ", format, args);
        }

        public static void Bug(string format, params object[] args)
        {
            MessageConsole(ConsoleColor.Red, "<BUG> ", format, args);
        }

        private static void MessageConsole(ConsoleColor color, string prefix, string format, object[] args)
        {
            Console.ForegroundColor = color;
            if (args == null || args.Length == 0)
            {
                Console.WriteLine(prefix + format);
            }
            else
            {
                Console.WriteLine(prefix + string.Format(format, args));
            }

            Console.ResetColor();
        }
    }
}
```

Figur 4: Klassen Dbg med funktionsdefinitioner.

6.3.7 Programflöde

När en användare skickar ett meddelanden till den virtuella assistenten via Telegram Messenger-klienten på till exempel deras mobiltelefon skickas det genom Microsoft Azure Bot Framework där det automatiskt konverteras till ett generellt format och skickas in till webbapplikationen *ExjobbBot*, programflödet följer pilarna i figur 3. När webbapplikationen tagit emot ett meddelande från en användare skickas meddelandet till LUIS-instansen, *Cognitive ExjobbBot*, som skickar tillbaka en tolkning av meddelandet till webbapplikationen baserat på inlärda tolkningar av meddelanden. Utifrån svaret från LUIS-instansen konstruerar webbapplikationen ett svar som därefter skickas tillbaka till användaren, potentiellt skapas också personuppgifter som i så fall sparas i *ExjobbBot User Database*. Den virtuella assistenten kommunicerar dessutom med ett API för tillgång till en extern produktdatabas, *Product Database* i figur 3, för att inhämta eventuella produkter att skicka tillbaka i responsen till användaren.

6.4 Utveckling av ExjobbBot

Programmet är uppdelat i ett flertal delar som uppfyller mindre uppgifter. Ett sådant exempel är initieringsdelen, som i figur 5 skapar den virtuella assistenten.

```
services.AddBot<ExjobbBot>(options =>
{
    options.CredentialProvider = new SimpleCredentialProvider(endpointService.AppId, endpointService.AppPassword);
    options.ChannelProvider = new ConfigurationChannelProvider(Configuration);
    ILogger logger = _loggerFactory.CreateLogger<ExjobbBot>();

    options.OnTurnError = async (context, exception) =>
    {
        Dbg.Bug("Uncaught Exception: {0}", exception.ToString());
        await context.SendActivityAsync("Sorry, it looks like something went wrong.");
    };
});
```

Figur 5: Skapande av den virtuella assistenten.

6.4.1 Användning av användardatabas

Kommunikation med användardatabasen skedde genom ett SQL-API. Detta API var tillhandahållet av Azure Cosmos DB, vilket var databastjänsten som användes för användardata. En projektmall med kod för att kunna ansluta och arbeta med databasen fanns tillgängligt för nedladdning efter databas-resursen hade skapats i Azure. Det enda som behövde göras för att börja arbeta med databasen var att konfigurera nycklar och ändpunkt för att ansluta till instansen av databasen. Under utvecklingen användes en lokal ändpunkt för databasen, då Azure Cosmos DB emulator fanns tillgänglig att köras lokalt. När detta var gjort var det bara att använda de tillhandahållna metoderna för att lagra och läsa data direkt från databasen.

6.4.2 Dialoger

Konversationer skapades genom att implementera Microsofts Bot Builder interfacet *IBot*, i klassen *ExjobbBot*. I konstruktorn sker initiering av datahanteringsklasser och registrering av dialogträd. *IBot* kräver att *OnTurnAsync* definieras, se figur 6, och det är den funktionen som körs varje gång användaren ger någon form av input till assistenten. Syftet med funktion är att kalla LUIS instansen *Cognitive ExjobbBot* över internet genom funktionen *RecognizeAsync*, vilket resulterar i en *RecognizerResult* som görs tillgänglig i alla dialoger. Resultatet innehåller *Cognitive ExjobbBot*s funna avsikter och entiteter i

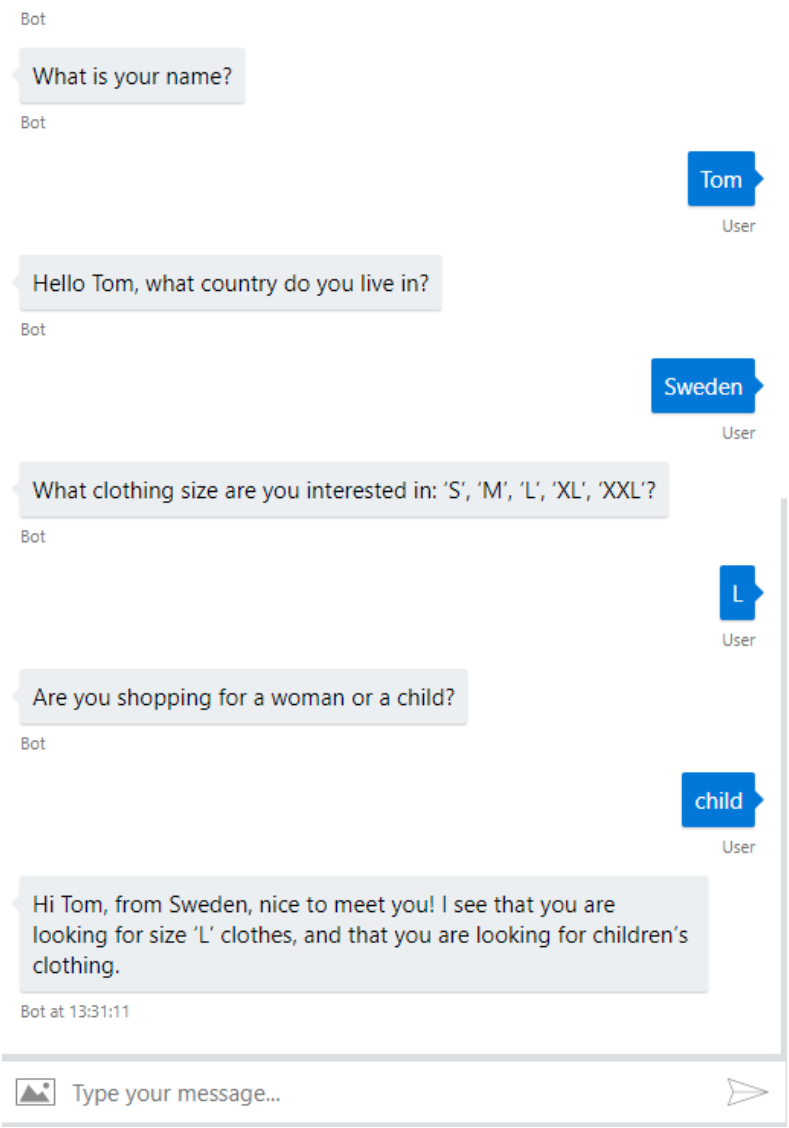
texten och dessa används för att undersöka om konversationen skall avbrytas, fortsätta på en tidigare dialog eller skapa en helt ny dialog.

```
var activity = turnContext.Activity;
string clientId = activity.From.Id;

// Create a dialog context
var dc = await Dialogs.CreateContextAsync(turnContext);

if (activity.Type == ActivityTypes.Message)
{
    // Perform a call to LUIS to retrieve results for the current activity message.
    RecognizerResult luisResults = null;
    try
    {
        luisResults = await _services.LuisServices[LuisConfiguration].RecognizeAsync(dc.Context, cancellationToken);
    }
    catch (APIErrorException)
    {
        Dbg.Warning("API Error: Unable to contact LUIS, likely due to too much traffic");
        await turnContext.SendActivityAsync("You're stressing me out! Stop right this instant!");
        return;
    }

    // If any entities were updated, treat as interruption.
    var topScoringIntent = luisResults?.GetTopScoringIntent();

    var topIntent = topScoringIntent.Value.intent;

    var humanState = await _humanStateAccessor.GetAsync(turnContext, () => new HumanState());
    var browseState = await _browseStateAccessor.GetAsync(turnContext, () => new BrowseState());
```

Figur 6: Partiell del av OnTurnAsync i ExjobbBot.

6.4.2.1 Klassen GreetingDialog

Denna dialog presenteras för användaren när den virtuella assistenten härleder att användaren skrivit en hälsningsfras. Dialogen ansvarar för att samla in personlig information från användaren, detta inkluderar namn, land, storlek på kläder som skall visas, samt om användaren vill visa kläder för kvinnor eller barn. Detta åstadkoms genom att dialogen i flera steg ställer de nödvändiga frågorna och inväntar svar samt sparar undan denna information i användardatabasen. Hur en full dialog av denna typ ser ut visas i figur 7.



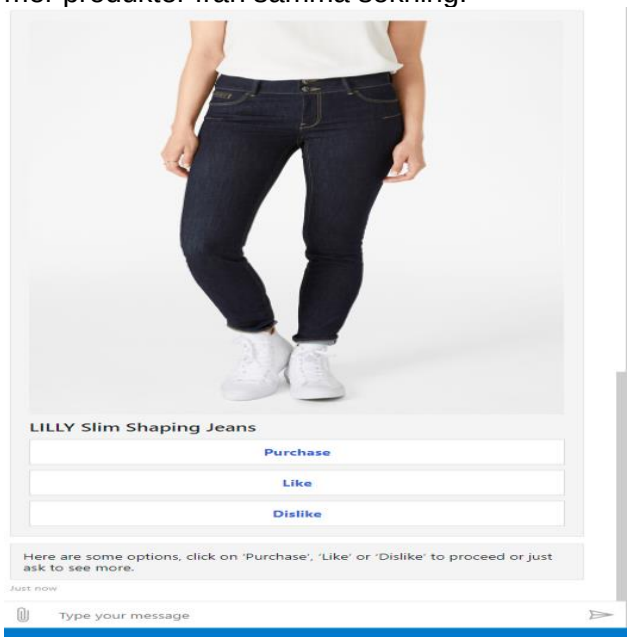
Figur 7: Exempel på en greeting-dialog.

6.4.2.2 Klassen BrowseDialog

När användaren är ute efter att hitta produkter kallas *BrowseDialog*; klassen som bygger ut klassen *ComponentDialog*. Konstruktorn i *BrowseDialog* skapar och registrerar alla olika vattenfallsdialoger och textpromptar som kan användas när användarens avsikt är att hitta produkter; detta genomförs genom att kalla funktionen *AddDialog*. I en standard implementeringen kallas dessa dialoger automatiskt i ordning men i implementeringen av assistenten ändras dock denna ordningen dynamiskt av förenklingsskäl.

När initieringsdialogen är genomförd byter den dialogordningen genom att lägga *ShowItemsDialog* först i dialogkön, detta resulterar i att funktionerna som är registrerade med identiteten *ShowItemsDialog* aktiveras och funktionen *ShowItemsAsync* kallas först. *ShowItemsAsync* undersöker först om användarens inmatning gick att tolka till produktattribut, och om det inte gick att tolka till attribut så skapar assistenten rekommendationsattribut. Resulterande attribut används därefter till att göra en sökning i produktdatabasen genom att kalla funktionen *GetProductsAsync* i klassen *ProductQuery*. Detta resulterar vanligtvist i att produkter returneras, och om inga produkter finns svarar assistenten med en kommentar om detta.

När produkter hittas plockar assistenten ut webbadresser till produktens största bild, vilken används för att bygga upp ett kort som skickas till användaren där de ser bilden och har möjlighet att klicka på knappar med feedback om produkten, se figur 8. I detta läge väntar assistenten på feedback, speciellt hanteras *IntentDislike*, *IntentLike*, *IntentBuy* och *IntentBrowseSeeMore*. Avsikterna om att gilla, ogilla eller köpa produkten resulterar i att assistenten sparar denna informationen som en *ProductSearch* i dokumentdatabasen, under kollektionen *Searches*, medan *IntentBrowseSeeMore* visar mer produkter från samma sökning.



Figur 8: Användarfeedback, bilden har beskurits.

6.4.3 Användning av produktdatabas

När programklassen *ProductDatabase* hämtar information från den externa produktdatabasen sker detta genom en mängd hjälpfunktioner som omvandlar all information till en HTTP-GET-adress och därefter vidarebefordrar denna till funktionen *DatabaseGetAsync*, se figur 9, vilken i grunden fungerar på samma sätt som ett CURL-kommando.

```
private static async Task<TResponse> DatabaseGetAsync<TResponse>(string url)
    where TResponse : class
{
    var request = (HttpWebRequest)WebRequest.Create(new Uri(url));
    request.Method = "GET";
    request.AllowAutoRedirect = false;
    request.ContentType = "application/json";
    request.Accept = "application/json";
    WebResponse response = null;
    try
    {
        response = await request.GetResponseAsync();
    }
    catch (Exception)
    {
        Dbg.Warning("Network error");
        return null;
    }

    if (response == null || ((HttpWebResponse)response).StatusCode != HttpStatusCode.OK)
    {
        Dbg.Warning("Unknown Server || Network || Format issue");
        return null;
    }

    using (var src = response.GetResponseStream())
    {
        string str2 = new StreamReader(src).ReadToEnd();
        List<string> errors = new List<string>();
        TResponse result = JsonConvert.DeserializeObject<TResponse>(str2, new JsonSerializerSettings
        {
            Error = (sender, args) =>
            {
                errors.Add(args.ErrorContext.Error.Message);
                args.ErrorContext.Handled = true;
            },
            Converters = { new IsoDateTimeConverter() },
        });

        if (errors.Count() > 0)
        {
            Dbg.Bug("Unable to parse JSON with following error(s):");
            foreach (string s in errors)
            {
                Dbg.Bug(s);
            }

            return null;
        }

        return result;
    }
}
```

Figur 9: Hämtning och formatering av data från produktdatabasen.

Detta sker genom att den skapar ett *HttpWebRequest* till den url som skickas in som parameter och därefter konverteras svaret till template-typen *TResponse*. *TResponse* skall vara definierad med samma fält som det JSON svar som skickas från produktdatabasens API, se figur 10 för ett exempel på *TResponse* vid en produktsökning, och dessa fält fylls i automatiskt av Newtonsofts JSON-konverterare.

```
public class SearchResponse
{
    public bool hasError { get; set; }
    public IList<string> querySuggestions { get; set; }
    public int totalHits { get; set; }
    public bool exactMatch { get; set; }
    public bool spellingCorrected { get; set; }
    public IList<Facet> facets { get; set; }
    public IList<Product> products { get; set; }
    public bool redirectToProduct { get; set; }
    public bool redirectToUrl { get; set; }
    public string redirectUrl { get; set; }
}
```

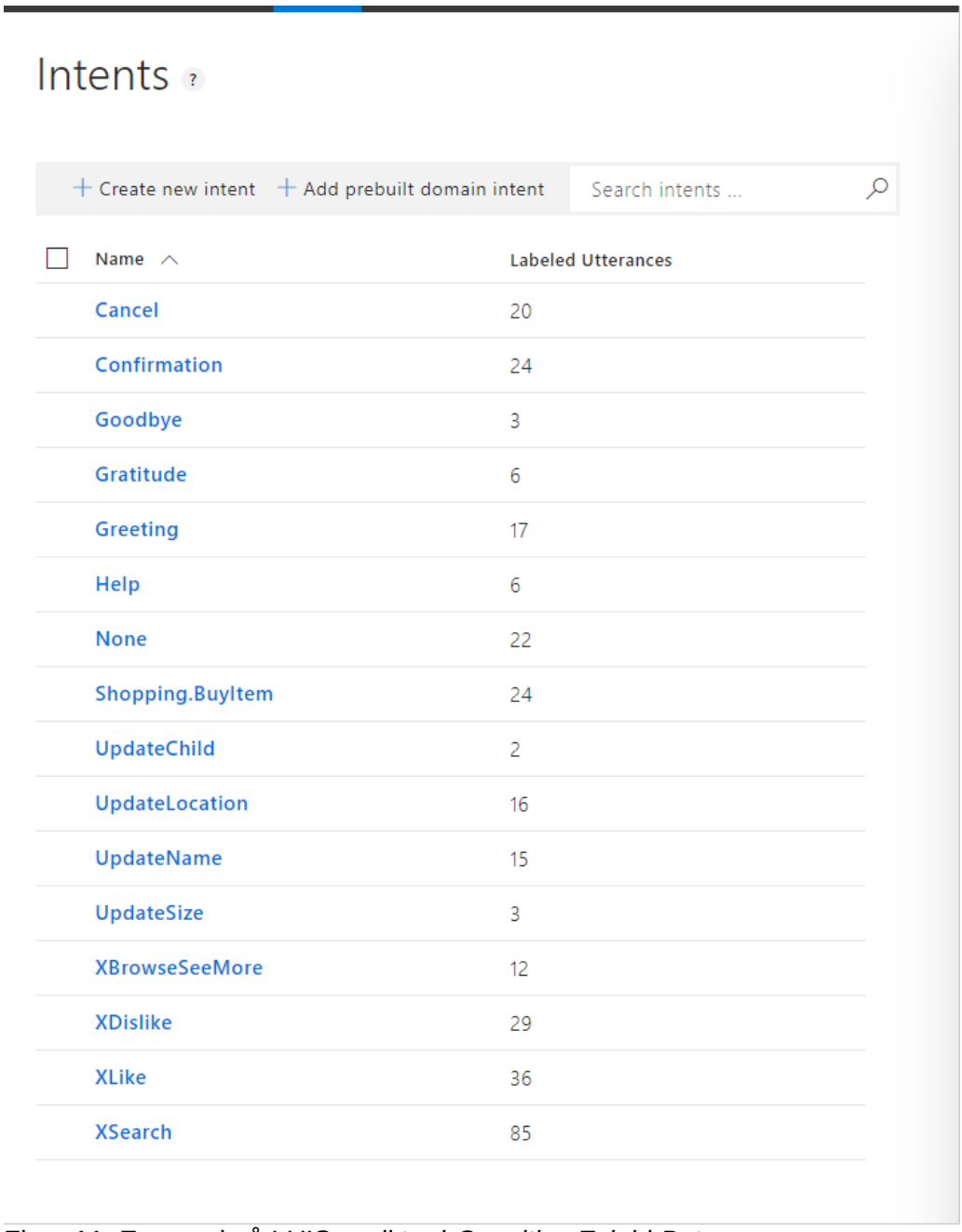
Figur 10: Svarsstruktur från produktdatabasen.

6.4.4 Lagring av användardata

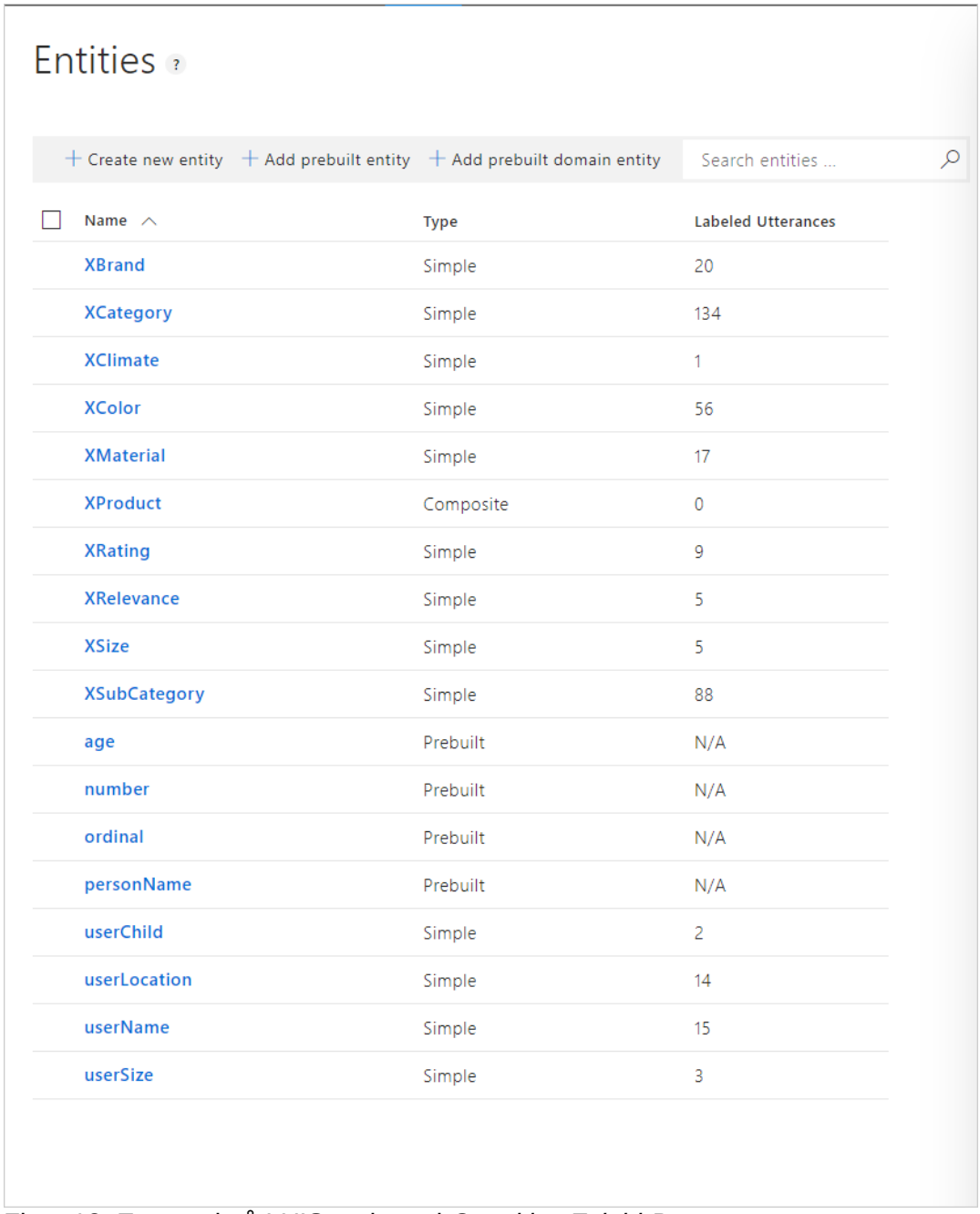
Den typ av information som samlas in från användaren är produktsökningar och användarens intresse, inklusive köp, eller avsaknad av intresse till en specifik produkt. Specifikt samlas klädesplagg, färg, typ av åsikt, material och id för en produkt. Denna information samlas in i en instans av klassen *ProductSearch*. Detta objekt omvandlades sedan till ett JSON-objekt och skickas för att sparas i användardatabasen med hjälp av API:et för Azure Cosmos DB.

6.5 Upplärning av Cognitive ExjobbBot

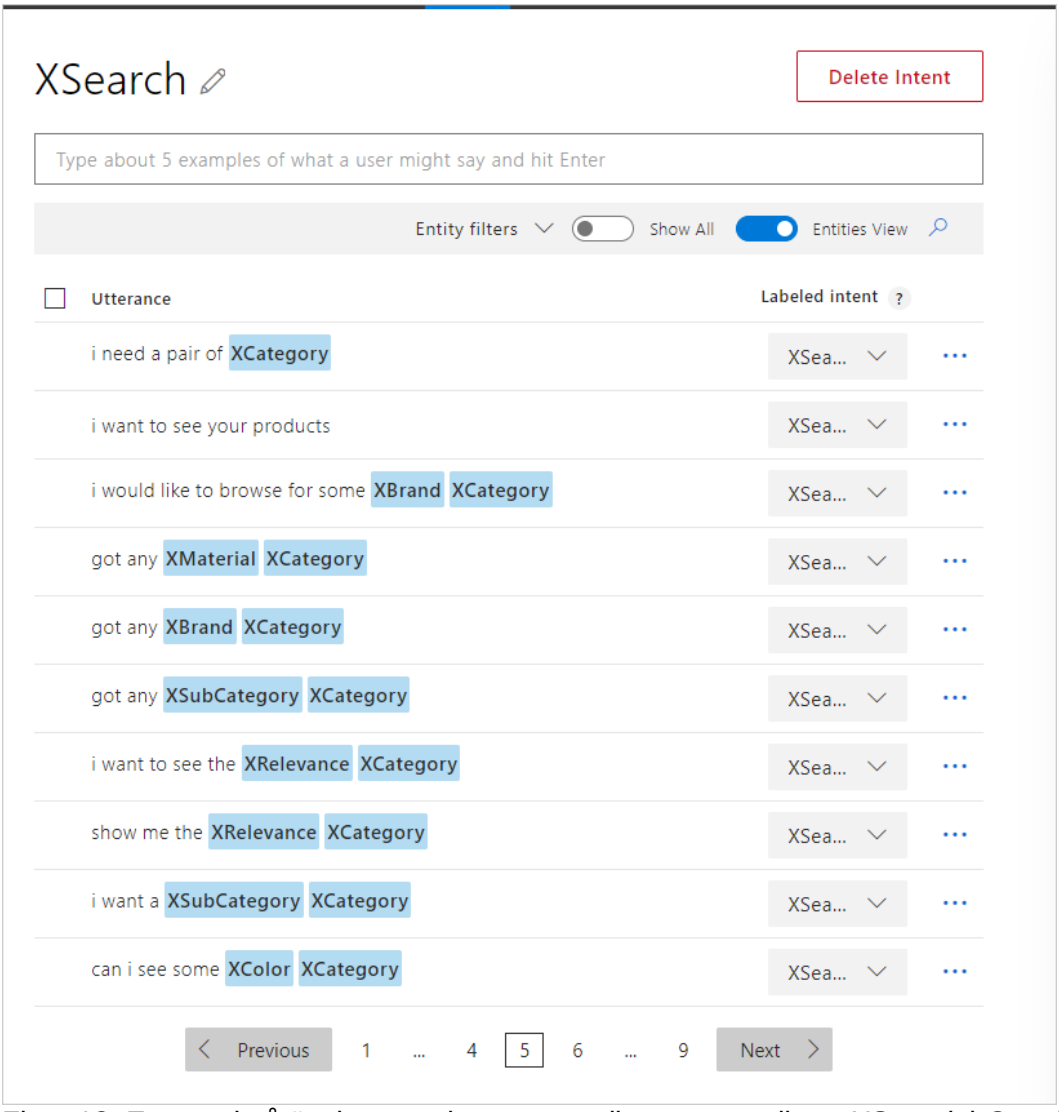
Instansen av LUIS konfigurerades med ett flertal avsikter, exempel på dessa finns i figur 11. Dessa avsikter valdes specifikt för att tillåta användaren att förmedla en sådan avsikt, till exempel *XSearch* är den avsikt som LUIS skall hitta när användaren ber om att få se produkter, alternativt *XBrowseSeeMore* när användaren vill se mer av en tidigare sökning etc. LUIS gavs också en lista på olika entiteter som skulle kunna finnas inne i avsikterna, exempel på dessa entiteter ses i figur 12. De entiteter som valdes att ta med var specifikt de som skulle vara användbara för VA, exempel på inlärning av avsikter och entiteter finns i figur 13. Efter att Cognitive ExjobbBot fick exempel på avsikter klickades *Train*-knappen för att träna upp LUIS och därefter klickades *Publish*-knappen för att publicera ändringarna i NLU AI:et.



Figur 11: Exempel på LUIS avsikter i *Cognitive ExjobbBot*.



Figur 12: Exempel på LUIS entiteter i *Cognitive ExjobbBot*.



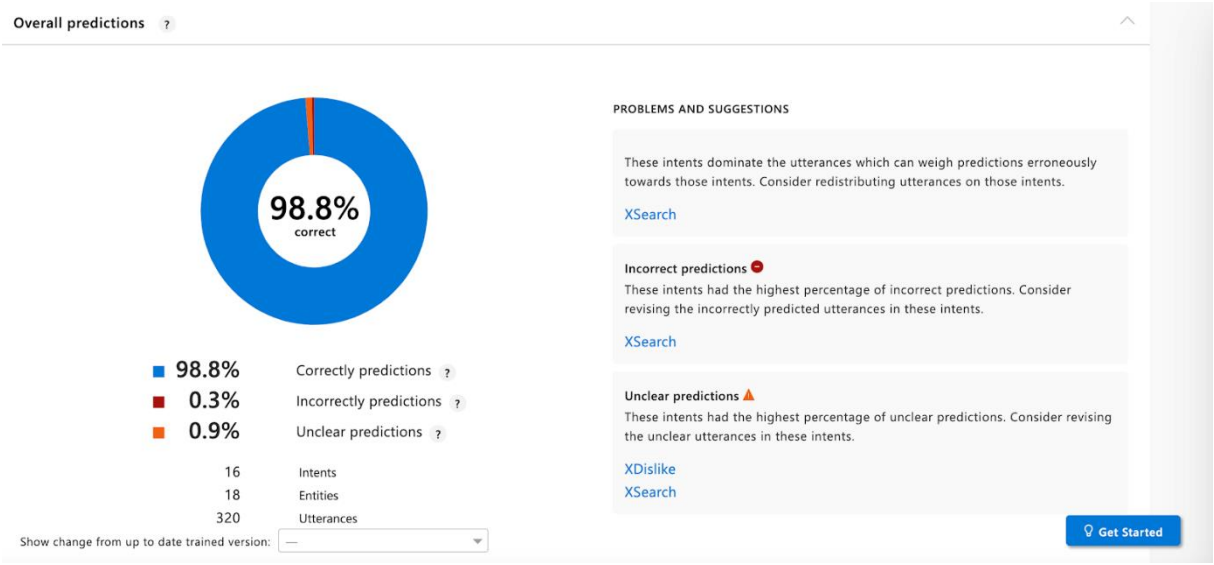
Figur 13: Exempel på övningsmeningar som tolkas som avsikten *XSearch* i *Cognitive ExjobbBot*, där blåmarkerade ord är entiteter.

7 Resultat

Projektet resulterade i en virtuell assistent specialiserad för e-handel. Den virtuella assistent som resulterade från detta projekt är kapabel till att ge användaren personliga produktrekommendationer, och behandla vanliga fraser från användaren, vilket var målet med projektet. Dessa översiktliga mål bestod dock av flera delmoment och funktionalitet. Bland annat behövde den virtuella assistenten kunna stödja en någorlunda normal konversation med användaren, och vara kapabel till att samla in användardata från dess konversationer med användaren. Detta var viktigt för att assistenten skulle vara lättanvändlig, effektiv och kapabel till att kunna ge personifiering av rekommendationer. I detta avsnitt beskrivs mer specifikt vad VA:n är kapabel till och hur detta åstadkoms. I tabell 1 nedan visas även exempel på fraser som VA:n behandlar korrekt och några som den inte helt behärskar. I figur 14 under tabellen visas även statistik på hur väl språkförståelse-komponenten har bedömt olika inlärningsfraser, men i verklig användning med nya fraser blir andelen korrekta tolkningar mycket mindre.

Fras	**Korrekt avsikt**	**Bedömd avsikt**	**Önskade entiteter**	**Hittade entiteter**
Show me some jeans	Sök	Sök	Kategori: jeans	Kategori: jeans
I'd like to browse some yellow shirts please	Sök	Sök	Färg: yellow Kategori: shirts	Färg: yellow Kategori: shirts
Hi bot I want to see jackets	Sök	Sök	Kategori: jackets	Kategori: jackets Namn: jackets
Greetings bot show me clothes	Sök	Hälsning		
My name is actually John	Byt namn	Byt namn	Namn: john	Namn: john
If you please could display some brown shoes it would be great	Sök	Sök	Färg: brown Kategori: shoes	Färg: brown Kategori: shoes
Hey there bot how are you?	Hälsning	Hälsning		
I actually want to shop clothes for children right now	Byt till barn	Byt storlek	Barn: children	
Howdy bot how are you today?	Hälsning	Hälsning		Namn: howdy bot
Update my profile to shop for size XL clothes	Byt storlek	Byt storlek	Storlek: xl	
I'm looking for some winter jackets today!	Sök	Sök	Kategori: jackets Klimat: winter	Kategori: jackets
I'm looking for some summer jackets today!	Sök	Sök	Kategori: jackets Klimat: summer	Kategori: jackets
Thank you very much for your assistance	Tack	Tack		
I like the black jacket	Gilla	Gilla	Färg: black Kategori: jacket	Färg: black Kategori: jacket
I really do not like that red gucci shirt	Ogilla	Ogilla	Färg: red Kategori: shirt Märke: gucci	Färg: red Kategori: shirt Märke: gucci

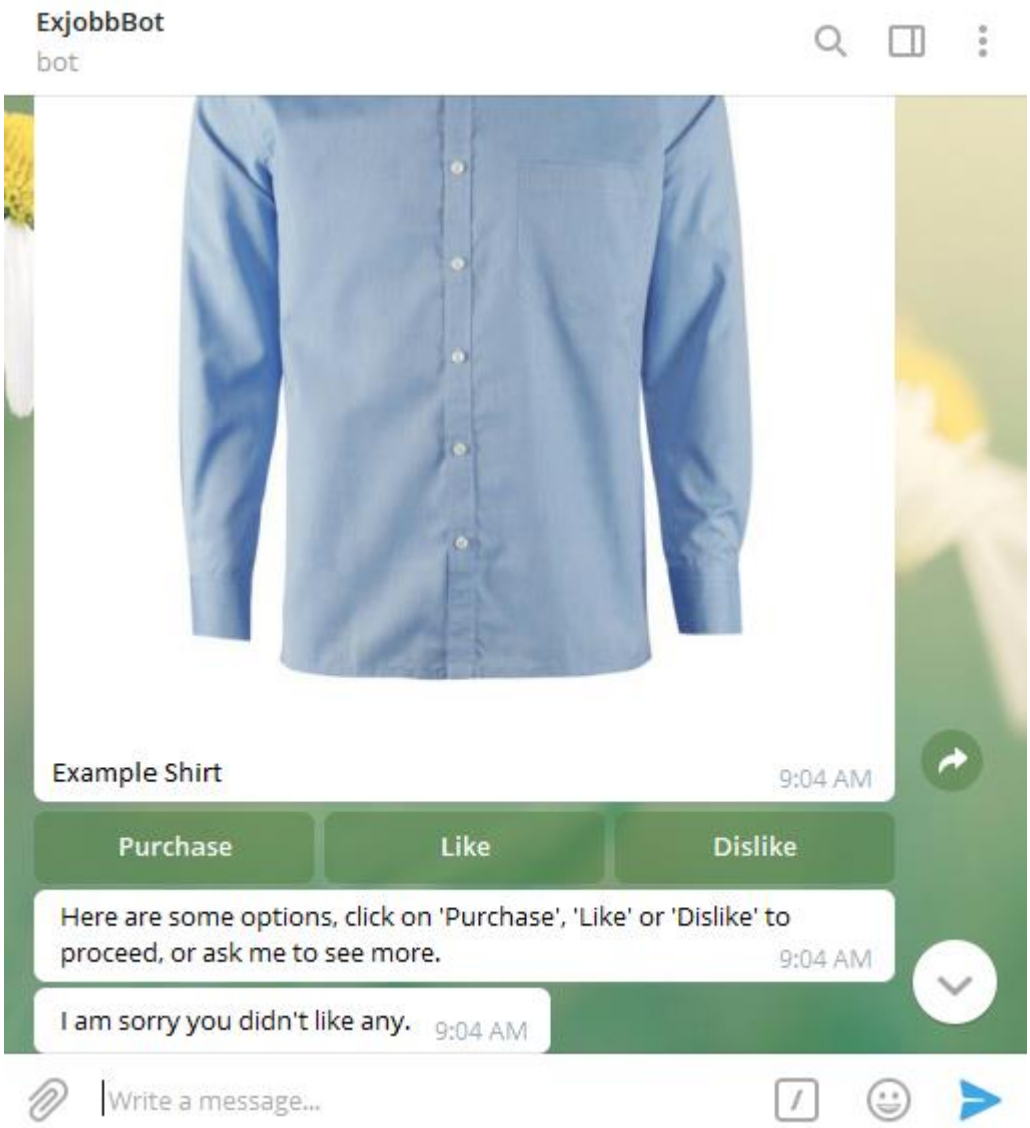
Tabell 1: Exempel på fraser som bedömts av språkförståelse-komponenten.



Figur 14: Statistik på hur väl språkförståelse-komponenten har agerat under användning.

7.1 Användningsfall

Den virtuella assistenten har utvecklats till att kunna fullfölja ett flertal möjliga användningsfall där assistenten ger användbar och i många fall specialiserad feedback för att hjälpa användaren vidare, och under hela processen sparas information för att senare kunna ge bättre rekommendationer. Det huvudsakliga användningsfallet som implementerades var att förstå när en användare vill söka efter en viss produkt, och följaktligen visa användaren produkter som passar in på sökningen och tolka återkoppling angående produkterna. En typisk interaktion visas i figur 15 där användaren har bett om att få se skjortor och därefter gett negativ återkoppling. Andra användningsfall som implementerades var exempelvis att förstå hälsningsfraser, bekräftelser och uppdatering av användardata.



Figur 15: En typiskt interaktion med den virtuella assistenten.

7.1.1 Hälsning

När den virtuella assistenten får en hälsning från användaren, och ingen profil har sparats för användaren än, går den vidare genom att fråga efter användarens namn och var användaren bor, om användaren är ute efter kläder till sig själv eller om den är ute efter kläder till dess barn, samt vilken storlek på kläder som eftersökes. All denna data används sedan när rekommendationer tas fram från sökningar av produkter. VA:n är kapabel att känna igen ett flertal förekommande hälsningsfraser, exempelvis “Hi”, “Hey”, “Greetings”, “What’s up?” och olika variationer av dessa, såsom “Hi there”, “Hey, how are you?”, “What’s up bot?”.

7.1.2 Uppdatering av användarinformation

Användaren kan också direkt ge information till assistenten genom att nämna informationen i form av ett påstående. Detta möjliggörs genom den språkförståelse som den virtuella assistenten har implementerats med. Informationen som ges på detta sätt tolkas på samma sätt som vid introduktionen. VA:n är tränad att urskilja både avsikten att uppdatera profildata, vilken profildata användaren vill uppdatera och vad användaren vill uppdatera den specifika användarinformationen med. Denna funktionalitet är inte

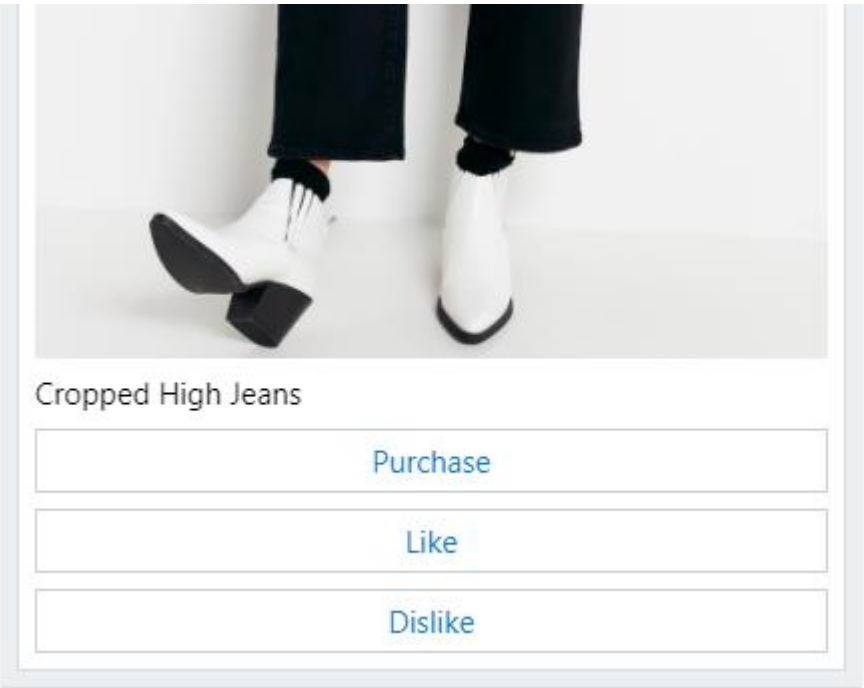
perfekt, precis som för alla andra användningsfall, men den känner exempelvis igen fraserna “I want to update my name to John” och “My name is actually John”. I dessa fraser är det namnet som är den specifika användarinformation som urskiljs vilja uppdateras, och “John” är informationen som användaren vill uppdatera det nuvarande värdet med.

7.1.3 Sökning av produkter

Den virtuella assistenten kan användas för att söka efter produkter genom att användaren ger en efterfråga om att se produkter. En efterfråga kan prydas med ytterligare information om vad användaren vill se, och den virtuella assistenten är kapabel till att tolka storlekar, färger, material och övriga kategorier, vilket sammanställs och används genom att enbart hämta de produkter som matchar användarens efterfråga. Slutligen sparas denna information i användardatabasen så att rekommendationerna får möjlighet att baseras på tidigare sökningar. Att användaren vill söka efter produkter urskiljs på samma sätt som beskrivet ovan, där avsikten och diverse information om vilken typ av produkter användaren vill se urskiljs och används för att söka i produktdatabasen. Exempel på sådan information är färg, material, märke och typ av klädesplagg. Förutom denna urskiljda information används den sparade användarinformationen, där storlek på kläder och om användaren efterfrågar damkläder eller barnkläder återfinns. Sparad data från tidigare sökningar används dessutom för att personifiera sökningen, genom att returnerade artiklar från sökningen sorteras och visas för användaren i den ordning som bäst återspeglar vad användaren visat intresse för tidigare. VA:n är kapabel att känna igen vanliga fraser för sökning av produkter, exempelvis “Show me red dresses”, “Can you please show me some green jackets?”, “I want to browse brown shoes please” och variationer av dessa.

7.1.4 Återkoppling från användaren

Användaren har möjlighet att ge återkopplingen och få ytterligare svar av den virtuella assistenten genom att antingen klicka på de knappar som ges med varje bild som skickas, se figur 15. Dessa knappar fungerar som en genväg från att skriva ut ett helt påstående angående om de tycker om produkten eller inte, samt om de är intresserade av att köpa produkten. Vid klick på ‘Purchase’ eller sändning av ekvivalent kommentar svarar assistenten med en länk till där du kan köpa produkten. Vid klick på ‘Like’, ‘Dislike’ eller sändning av snarlik kommentar kommer assistenten svara genom att påpeka att den kommer komma ihåg användarens återkoppling; vilket senare används i produktrekommendationer för användaren. Eftersom flera produkter kan visas samtidigt kan det bli problematiskt för assistenten att veta specifikt vilken produkt användaren gav kommentar om, vilket användning av de standardiserade knapparna avhjälper. Vid problem med tolkningen kommer assistenten istället ge tillbaka de alternativ den hade problem att välja mellan; vilket ger användaren möjligheten att ge en mer specifik kommentar för att filtrera ut de felaktiga produkterna. Dessa kommentarer kan vara ospecifika såsom en kommentar som påpekar färgen eller tyget på det önskvärda klädesplagget, eller så pass specifika att de påpekar ID-numret på produkten.



Figur 16: Genvägsknappar för användaren.

7.1.5 Rekommendationer

Personliga produktrekommendationer skapas i två steg. I det första steget hämtades alla sparade sökningar och gillade produkter från användardatabasen. Denna data används sedan för att ta fram hur många gånger olika produktegenskaper förekommer i den sparade användarinformationen; såsom färg på klädesplagg, klädmärke och material. I det andra steget utnyttjades de framräknade värdena från det första steget till att sortera alla returnerade produkter från en viss sökning utefter deras egenskaper. Om exempelvis en användare enbart söker på jeans hämtas tjugo produkter relaterade till det ordet från produktdatabasen. De hämtade produkterna sorteras därefter baserat på användarens historik på så sätt att produkter med egenskaper som förekommer oftast i användarens tidigare sökningar och gillade produkter hamnar längst fram i listan av sorterade produkter jämfört med produkter som har egenskaper som förekommer mer sällan eller inte alls. Från denna sorterade listan kan sedan användaren skrolla genom produkterna, två produkter i taget, där de produkter som har bedömts att användaren troligen kommer att gilla mer visas först för denne och produkter som användaren ogillar inte visas alls. Anledningen till att produkterna enbart sorteras och assistenten inte enbart visar de produkter med högst prioritet är att användaren skulle då inte få chansen att hitta nya produkter.

7.1.6 Generell felhantering

Problem kan uppstå om användaren skickar in en kommentar som inte passar in på vad den virtuella assistenten var ute efter. I sådana problematiska fall kommer assistenten be användaren att förtydliga vad den syftade på, varpå användaren kan förtydliga eller alternativt avsluta den delen av konversationen; vilket tillåter användaren att återkomma till början av konversationen. Om den virtuella assistenten blir tvungen att hantera mer konversationstrafik än den använda versionen av LUIS tillåter, återkommer assistenten med kommentaren att den är stressad och på så sätt antyder den därav att användaren kanske inte kommer få svar på sina frågor för tillfället.

8 Slutsats

Projektet resulterade i ett komplett program kapabelt att göra det som angavs i projektets mål, nämligen att med användning av nya tekniker som AI skapa en virtuell assistent för kundsupport inom e-handel, och att behandla några vanliga dialoger, exempelvis hälsningar, uppdatering av profil, och sökning av produkter. AI-tekniker användes i projektet enbart för språkförståelse, medan mer konventionella metoder användes för att sedan ta fram de personliga rekommendationerna. Det var först tänkt att möjligtvis även applicera någon AI-baserad metod på att strikt ta fram rekommendationerna också, men som konstaterades under avsnittet Teoretisk bakgrund ersattes denna lösning med en annan. Med hänvisning till ovan nämnda resultat bedöms målen med projektet mött. Konkret kan slutsatsen dras att användning av molnbaserade AI-tjänster effektivt går att använda till att bygga VA:er till e-handel.

9 Diskussion

Överlag var projektet lyckat. Det mål som angavs i början av rapporten anses ha mötts, även fast det hade gått att göra saker annorlunda. VA:en är dock inte kapabel att känna igen alla möjliga variationer av olika fraser tillhörande de avsikter som implementerades, exempelvis behandlar inte VA:en meningen "*My saved name is wrong, it's actually John*", där avsikten att uppdatera profilen urskiljs, men namnet som användaren vill uppdatera till bedöms vara "*wrong*". Det finns massvis med liknande fall där VA:en inte urskiljer avsikt eller tillhörande entiteter korrekt. I samtliga fall ser man betydligt färre felbedömningar på fraser som är ämnade för avsikter med många olika variationer av exempel fraser inträngade, medan de fraser som tillhör en avsikt som har betydligt färre exempel specificerade och inträngade behandlas felaktigt med högre frekvens. Detta är förståeligt och väntat då det är så neurala nätverk fungerar; mängden data som används och relevanta variationer i denna är viktiga egenskaper för att nå önskvärda resultat.

En sak som hade varit intressant att göra annorlunda hade varit att använda sig av AI-baserade lösningar för att sammanställa de personliga rekommendationerna till användarna. Under Teoretisk bakgrund beskrivs ett spontant förslag på hur detta möjligen hade kunnat gå till, även fast förslaget är ofullständigt för projektets ändamål. Som konstaterades under avsnitten Resultat och Slutsats valdes istället en något enklare, men fortfarande effektiv, lösning för de personliga rekommendationerna. Det går dessutom att använda olika AI-baserade metoder för ändamålet att ta fram personliga rekommendationer. Det går exempelvis att träna ett artificiellt neuralt nätverk på de olika attributerna hos kläderna, såsom färg, klädesplagg, märke och material, men det hade även gått att träna ett nätverk direkt på de visuella likheterna i bilderna av kläderna.

Även fast Telegram Messenger användes i detta projekt, går det att integrera den virtuella assistenten med andra meddelandetjänster som exempelvis Facebook Messenger och Slack. Det hade alltså gått att koppla upp till ett flertal meddelandetjänster för att på så vis exponera VA:en till fler användare.

9.1 Samhälls- och miljöpåverkan

VA:er möjliggör att kundtjänst hanteras av datorer istället för människor, då mänsklig arbetskraft har en stor miljöpåverkan kan det således förbättra utvecklingen för miljön om arbete effektiviseras till en grad där människor spenderar minimal arbetstid för en viss arbetsuppgift. Trots att dessa VA:er kan vara negativa för de individer som i nuvarande läge ansvarar för kundtjänst anser vi att i helhet bör effektivisering prioriteras över bevarande av föråldrade arbetstitlar.

Vid utveckling av program med användardata finns vissa risker för användares integritet, då denna information kan potentiellt användas till olagliga eller oönskade ändamål. Användaren bör informeras hur användarinformationen sparas samt vilken användarinformation som sparas och hur den kan komma att användas. Utförandet av detta projekt skedde inte i syfte att hantera användarinformation på ett säkert sätt och användarinformation har därför sparats undan okrypterat, vilket gör applikationen obrukbar för offentlig publicering enligt dataskyddsförordningen (DSF).

Källförteckning

[1] "Humany - Smartare kundservice på nätet", *Humany.se*, 2019. [Online]. Available: http://www.humany.se/. [Accessed: 24- Apr- 2019].

[2] Chatbots Magazine. (2019). *Chatbot Report 2018: Global Trends and Analysis – Chatbots Magazine*. [online] Available at: https://chatbotsmagazine.com/chatbot-report-2018-global-trends-and-analysis-4d8bbe4d924b [Accessed 14 Feb. 2019].

[3] 2019. [Online]. Available: https://www.mobil.se/nyheter/reportage/reportage-det-h-r-r-ai-artificiell-intelligens-och-maskininl-rning. [Accessed: 02- Apr- 2019].

[4] Unemyr, M. and Unemyr, M. (2019). *Automatisk segmentering och personalisering med AI - Magnus Unemyr Marketing Automation & AI Blog*. [online] Magnus Unemyr Marketing Automation & AI Blog. Available at: https://unemyr.com/ai-segmentering-personalisering/ [Accessed 22 Feb. 2019].

[5] 3bits.se. (2019). *3bits skapar effektiv e-handel - allt från webb till logistik. - 3bits*. [online] Available at: https://www.3bits.se/sv/ [Accessed 24 Feb. 2019].

[6] "Molntjänster – vad är molnet egentligen?", *Vismaspcs.se*, 2019. [Online]. Available: https://vismaspcs.se/ditt-foretagande/driva-eget-foretag/molntjanster. [Accessed: 02- Apr- 2019].

[7] 2."SQL vs NoSQL: The Differences — SitePoint", *SitePoint*, 2019. [Online]. Available: https://www.sitepoint.com/sql-vs-nosql-differences/. [Accessed: 20- Mar- 2019].

[8] "Microsoft Azure-dokumentation", *Docs.microsoft.com*, 2019. [Online]. Available: https://docs.microsoft.com/sv-se/azure/index#pivot=products&panel=all. [Accessed: 02- Apr- 2019].

[9] "Azure Bot Service Introduction - Bot Service", *Docs.microsoft.com*, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overview-introduction?view=azure-bot-service-4.0. [Accessed: 02- Apr- 2019].

[10] "Introduction to Azure Cosmos DB", *Docs.microsoft.com*, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/cosmos-db/introduction. [Accessed: 20- Mar- 2019].

[11] "Utveckla lokalt med Azure Cosmos-emulatorn", *Docs.microsoft.com*, 2019. [Online]. Available: https://docs.microsoft.com/sv-se/azure/cosmos-db/local-emulator. [Accessed: 02- Apr- 2019].

[12] "Test and debug bots using the Bot Framework Emulator - Bot Service", *Docs.microsoft.com*, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0. [Accessed: 02- Apr- 2019].

[13] "Vad är Language Understanding Intelligent Service (LUIS) – Azure Cognitive Services", *Docs.microsoft.com*, 2019. [Online]. Available: https://docs.microsoft.com/sv-se/azure/cognitive-services/luis/what-is-luis. [Accessed: 02- Apr- 2019].

[14] "Telegram F.A.Q.", *Telegram*, 2019. [Online]. Available: https://telegram.org/faq#q-what-is-telegram-what-do-i-do-here. [Accessed: 02- Apr- 2019].

[15] "Bots: An introduction for developers", *Core.telegram.org*, 2019. [Online]. Available: https://core.telegram.org/bots. [Accessed: 02- Apr- 2019].

[16] "Introduction - Messenger Platform - Documentation - Facebook for Developers", *Facebook for Developers*, 2019. [Online]. Available: https://developers.facebook.com/docs/messenger-platform/introduction. [Accessed: 02- Apr- 2019].

[17] "IBM Cloud Docs", *Cloud.ibm.com*, 2019. [Online]. Available: https://cloud.ibm.com/docs/services/assistant?topic=assistant-index#index. [Accessed: 02- Apr- 2019].

[18] Dialogflow. (2019). *Overview | Dialogflow*. [online] Available at: https://dialogflow.com/docs/intro [Accessed 9 Apr. 2019].