

Evaluation of Modern Tools for an OMSCS Advisor Chatbot

Eric Gregori

Georgia Institute of Technology
Atlanta, Georgia, USA
egregori3@gatech.edu

ABSTRACT

This paper is a survey of modern chatbot platforms, Natural Language Processing tools, and their application and design. A chatbot is proposed for the GA Tech. OMSCS program to answer prospective students questions immediately 24/7.

INTRODUCTION

The Georgia Institute of Technology OMSCS program is an online Masters of Computer Science program with over 4500 students across 99 countries as of Spring 2017. [7] The program is supported by a group of dedicated and overworked advisors who do their best to answers the potential and current students questions as soon as possible.

“Please allow 24-48 business hours as a standard response time. During certain times of the year, such as registration and the end of the term, the response time may increase. We greatly appreciate your patience and understanding” OMS CS Advising Team. [8]

The goal for this project is to reduce the response time from 24-48 hours to a few minutes using conversational chatbot technology.

A conversational chatbot is a tool that uses natural language processing (NLP) for human machine interaction (HMI). Historically, chatbots were used for entertainment. With the goal of tricking a human into thinking they were talking to another human. Modern chatbots are tools used to convey information.

“Chatbots, with their pattern matching, natural language processing, and ability to connect multiple patterns to one response, are a natural fit for providing reference and instructional assistance.” [2]

Investment from Amazon, Google, Apple, Microsoft, and Facebook has progressed NLP significantly. Driven by popular NLP based products: Siri, Alexa, Cortona, and Facebook. Google, Microsoft, and Facebook have made their technology available to developers via on-line API’s and/or open source libraries.

These tools have increased the quality of chatbots while at the same time decreasing the cost of entry resulting in a boon in the use of chatbots.

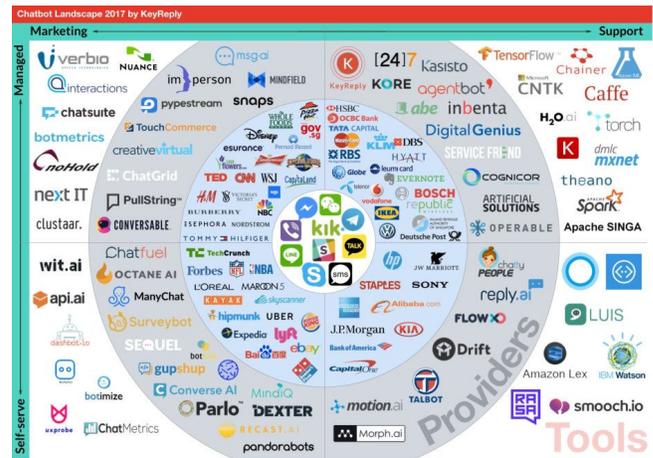


Image credit: Keyreply.com

COMMUNITIES THAT CAN BENEFIT FROM CONVERSATIONAL CHATBOT TECHNOLOGY

Conversational chatbot technology has the potential to benefit any community that requires interactive communications but does not have the resources to provide it manually. For example customer support and education.

“51% of people say business needs to be available 24/7.” [12]

“45.8 would rather contact a business through messaging than email.” [12]

“So what’s a brand to do in building its own chatbot?”

Here are three considerations to think through:

Design the chatbot to serve a purpose: A retailer will likely create a chatbot to help people purchase goods. A telco might create one to deal with customer service. Knowing the purpose of the chatbot means knowing which conversation threads are essential, and which might be extraneous to the design. In our experience, chatbots can be designed for four purposes: content distribution, customer service, recommendation, and transaction.

Approach it as one would a product design project: In a chatbot, content is what the user interfaces and engages with. Content design entails knowing the product opportunities and limitations that can or cannot be resolved by the chatbot. For example, a logistics company helping customers to track deliveries will need to provide access to

real-time package-location data. If this is not technically feasible for a chatbot, it should not be designed within the interface.

Have a beginning, a middle and an end: A purpose-driven chatbot will guide a user through to an intended end goal (conduct a transaction, serve a piece of content). Consider the journey a user will take when they encounter the bot, how the customer will move into an interaction and out of it. Off-script conversations are also inevitable, and good chatbot designs build in paths to navigate such conversations back on-track.” [11]

Customer Support

“In general, clients wait for a response for too long or even don’t get one. This leads to negative reviews and dissatisfied customers.” [14]

“German company Tec inStore had problems. They needed many workers to respond to the same questions, that’s why they were losing a lot of their customers.

Heads of Tec inStore decided to create a chatbot which can respond to typical questions of consumers. If it is necessary a bot can contact person to Call Center.

After one month of usage, chatbot had an 80% success rate, reduced requests to the customer support, more than 1500 people have requested help from the chatbot. Chatbot received more than 10 000 messages from which 1000 were the “Thank You” Messages.” [14]

Harvard Business Review coined the term “conversational commerce,” the term for online business that’s powered by natural language technologies. Using artificial intelligence (AI) technologies for relevant, personal, and helpful interactions with customers. AI-driven conversations is the ability for the brand to zoom in immediately on what customers need (regardless of how they say it), based on strong understanding of context. [18]

The capacity to recognize a user’s tone is an important aspect of good customer support. [15] Determining when a user is getting frustrated or angry and acting accordingly. Using sentiment analysis, a conversational chatbot can detect the user’s tone and use the information when generating a response. [17]

“Personalizing customer service. The potential to improve customer service while lowering costs makes this one of the most exciting areas of opportunity. By combining historical customer service data, natural language processing, and algorithms that continuously learn from interactions, customers can ask questions and get high-quality answers.” [19]

Harvard Business Review states, “44% of U.S. consumers already prefer chatbots to humans for customer relations.” [19] Chatbots can improve customer loyalty and retention. Sentiment analysis can be used to classify customer

satisfaction. Post analysis of the logged conversation can be used to improve the chatbot for future interactions.

Rob High from IBM stresses the importance of emotional intelligence in chatbot when used for customer support. *“The goal of every conversation with an AI-enabled chatbot should be to ensure the user feels satisfied and understood, even in sensitive situations.” [20]*

Accenture Interactive provides a guide to using chatbots for customer service. The guide lists four guidelines to factor when considering a chatbot for customer service applications: focused scope, multiple steps or input parameters, high volume of requests, and unstressed customers.[21]

Focused scope: chatbots are most effective for troubleshooting or customer education. Multiple steps or input: the natural language capabilities of chatbot makes it easier for user to convey multiple parameters quickly. High volume of requests:

Education

“Chatbots, with their pattern matching, natural language processing, and ability to connect multiple patterns to one response, are a natural fit for providing reference and instructional assistance.” [22]

ANTswers (<http://answers.lib.uci.edu/>) is a librarian chatbot at the university of California Irvine. ANTswers is based on the open source framework A.L.I.C.E. ANTswers was created because the library did not have the resources to support patrons 24/7. ANTswers is implemented as an interactive FAQ. [22]

Freudbot

(<https://psych.athabascau.ca/html/Freudbot/test.html>) is a chatbot programmed to chat in the first person about Sigmund Freud’s theories, concepts and biographical events. Freudbot is based on the open source framework A.L.I.C.E. Freudbot is a tool to study how students interact with chatbot technology. The study found the chatbot had a positive utility in on-line utility. The biggest complaint about Freudbot was the simplicity of the technology. This study was done in 2005. The technology has improved drastically in the last 12 years. [24]

AdmitHub “pounce”

(<http://blog.admithub.com/case-study-how-admithub-is-freezing-summer-melt-at-georgia-state-university>) is a custom virtual assistant for Georgia State University (GSU) admissions. [27]

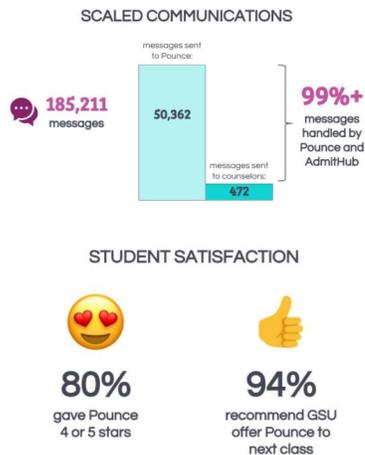


Figure 1. Pounce first year statistics [27]

As shown in figure 1, Pounce was very successful and GSU decided to continue using Pounce the next year. “In our focus groups with students who engaged with Pounce, we learned that some of the common themes were that students didn’t feel judged for asking what might seem like a “stupid” question, they appreciated the instantaneous responses, especially when they asked questions at all hours of the night ...” [27] AdmitHub’s research found that, “55% of millennials, reported that using a bot left them with a more favorable impression of the business.” [26]

The Pounce chatbot had a knowledge base of over a thousand FAQ’s. On average each student interacted with Pounce 14 times. [28]

An interesting use of chatbots in education is intelligent tutoring systems. “Pedagogical agents are autonomous agents that occupy computer learning environments and facilitate learning by interacting with students or other agents (e.g. [29], [30], [31], and [32]). They may act as peers, co-learners, competitors, helpers or instructors. For effective pedagogy, agents should be able to ask and respond to questions, give hints and explanations, monitor students and provide feedback [33].” [23]

VPINO is a coaching/counseling chatbot based on IBM’s Watson. “A text based natural language dialogue system specifically developed for the purpose of holding structured, goal directed coaching conversations. VPINO aims to simulate a human like conversation, including social cues, grounding techniques and human turn taking behaviour.” [34]

VPINO uses Socratic questioning, a general conversation technique that is attributed to the Greek philosopher Socrates. Socratic conversation is used to explore implicit knowledge, uncover assumptions and follow logical implications by systematic and disciplined questioning. [34]

OMSCS Community

The OMSCS community is made up of two distinct student types: current students and prospective students. All current students started as prospective students.

Spring 2017 Current students [7]

- Overall Enrollment: 4,515
- U.S. Citizens & Residents (% of Enrollment): 74.1%
- International Students: 25.9%
- Men: 86.1%
- Women: 13.9%
- Underrepresented Minorities: 14.7%
- Total Course Enrollments: 6,518
- Countries Represented: 99
- U.S. States/Territories Represented: 51
- Companies Represented: 2,500

As of spring 2017 there have been 13 thousand applications for the OMSCS program. [6] Not all prospective students apply. All current students submitted applications. Based on this limited data we can infer that the community of prospective students is at least 13 thousand people scattered around the world.

Prospective and current students ask different questions and are therefore supported by different knowledge bases. Current students have access to data behind the Buzzport login while prospective students are limited to the public online material.

Prospective students currently have three online resources for information: FAQ’s, reddit, and Google community.

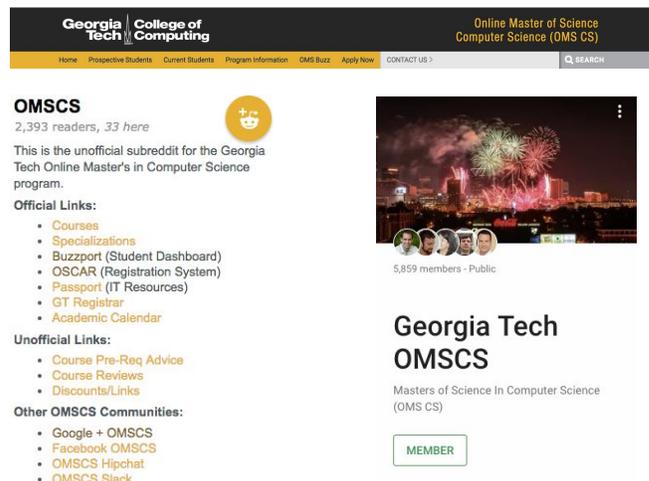


Figure 2. GATech FAQ (top), Reddit (left), and Google+ (right)

The OMSCS advisor team has done a great job of putting all the required information for prospective students on line.

The data is available, but there is a lot of it.

“It has been argued that the true value of dialogue interface systems over direct manipulation (GUI) can be found where task complexity is greatest [5]. Specifically, when used (a) to filter/browse or issue commands over large amounts of information/sets of things (b) to follow navigational paths (c) to allow negation and quantification (d) to distinguish individuals from kinds (e) to filter and request information in ways not predetermined by design, (f) to build a complex query that takes more than a single turn, (g) for referring to things that you cannot see or point to, or doing anything that’s not in the here and now, and (h) for delegating complex or redundant actions.” [4]

Based on this research, a chatbot is a good fit for this knowledge base and community.

OMSCS Corpus

GT Home > Home > Prospective Students

FAQ

[OMS CS Admissions](#)

[About the Degree Program](#)

[Cost and Transfer Credit](#)

[Why the OMS CS?](#)

[About the Collaborators](#)

[MOOCs, Education & Learning](#)

[The Need for STEM Education](#)

Figure 3. <http://www.omscs.gatech.edu/prospective-students/faq>

The OMSCS corpus is based on the FAQ shown in figure 3. A small portion of the FAQ is shown in appendix A. The text in appendix A is used for testing various chatbot systems.

CONVERSATIONAL CHATBOT TECHNOLOGY

“The aim of chatbot designers is to build tools that help people, facilitate their work and their interaction with computers using natural language, but not imitate the human conversation perfectly i.e. not to replace human role completely.” [1]

Domain

The complexity of the chatbot is directly proportional to the scope of the domain. An open domain requires a large vocabulary and knowledge base. A closed domain limits the vocabulary and knowledge base to the requirements of a specific goal. “In a closed domain (easier) setting the space of possible inputs and outputs is somewhat limited because the system is trying to achieve a very specific goal. Technical Customer Support or Shopping Assistants are

examples of closed domain problems. These systems don’t need to be able to talk about politics, they just need to fulfill their specific task as efficiently as possible. Sure, users can still take the conversation anywhere they want, but the system isn’t required to handle all these cases – and the users don’t expect it to.” [49]

Natural Language Processing (NLP)

“Natural language processing (NLP) is a theory-motivated range of computational techniques for the automatic analysis and representation of human language.” [48] NLP technology is progressing quickly from word or pattern matching (“Trying to ascertain the meaning of a piece of text by processing it at word level, however, is no different from attempting to understand a picture by analyzing it at pixel-level.” [48]) to advanced machine learning based systems capable of extracting intent or meaning from natural language.

Sentence or Utterance

A sentence is a group of words that convey a complete meaning or thought. An utterance is a part of speech between pauses and silence. Many modern NLP systems support both text and voice input and thus use the term utterance for both text and voice input.

Part-of-Speech (POS) Tagging

“Part-of-speech (POS) tagging involves labeling each word in the input with a tag that indicates its syntactic role ...” [3] POS tagging is the first step in NLP.

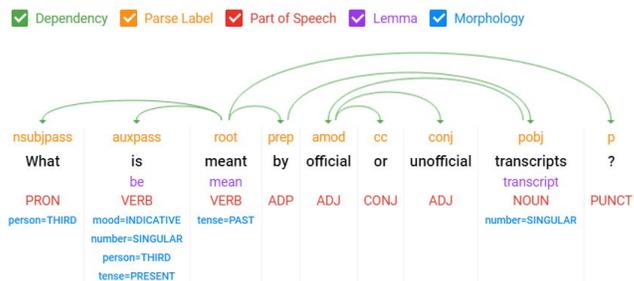


Figure 4. POS Tagging

Figure 3 shows the sentence, “What is meant by official or unofficial transcripts?” POS tagged by Google Natural Language (appendix B).

Entity Recognition

“The first step in information extraction is to detect the entities in the text. A named entity is, roughly speaking, anything that can be referred to with a proper name: a person, a location, an organization. The term is commonly extended to include things that aren’t entities per se, including dates, times, and other kinds of temporal expressions, and even numerical expressions like prices. Named entity recognition means finding spans of text that constitute proper names and then classifying the type of the entity.” [52]

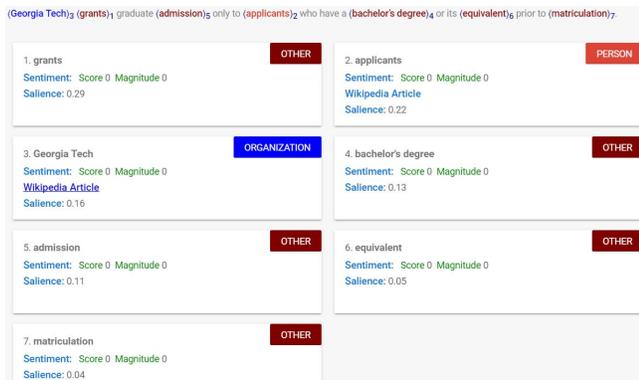


Figure 5. Entity Recognition/Classification

Figure 5 is an entity analysis performed by Google Natural Language (appendix B). The entities are recognised and classified. Some entities are not classified correctly because the classifier has not been trained to the domain yet.

Intent Analysis

At the highest level of the NLP stack is Natural Language Understanding (NLU). Using the image analogy, while NLP is at the pixel level, NLU is at the image level. NLU algorithms attempt to infer meaning from sentences classifying groups of words as opposed to individual words.

Intent analysis (IA) is a NLU algorithms for decoding the user's intention from a sentence or utterance. Modern IA algorithm use various machine learning techniques to classify sentences into intents. "Intents are the intentions of the end-user." [47]

The IA algorithm must be trained to categorize sentences into intents. A list of sentences with similar intent (but worded differently) is passed to the algorithm and assigned to a tag (intent).

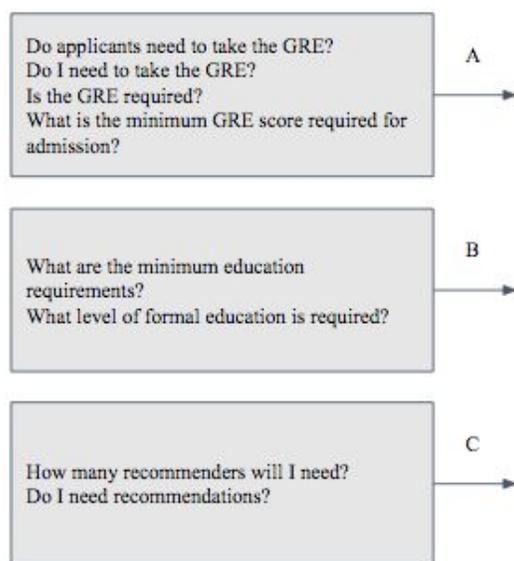


Figure 6. Grouping utterances into intents A,B, and C

Figure 6 shows three intents, their utterances, and the resulting output. For group A, the users intent is to determine if a 'GRE' is 'required'. Not to determine 'where' to take the 'GRE'. This is an example of how intent classification differs from simple entity classification (which would just use the entity 'GRE').

Sentiment Analysis

"Sympathy is a key feature in the way we impart. It's one reason why we get so baffled at endless telephone menus when we call client administration. We simply need to converse with a man! On the off chance that we could just converse with a real person they would comprehend the issue we're having [16]. They would know how disappointed we are by the words we use, as well as by our articulation." [15]

Using a database of sentiment classified words (sentiment lexicon) the user's input is classified as positive or negative. The chatbot uses the classification to create its response and possibly when to end the conversation and escalate the conversation to a human. [17]

Dyadic Conversation

Up until very recently, chatbots have only supported a single adjacency pair. This is referred to as a one-shot conversation. Siri, Alexa, and Google assistant are examples of speech based one-shot conversation systems. On the leading edge are chatbots that can support multiple adjacency pairs. Remembering states between pairs and capable of associating data in different adjacency pairs (being able to maintain the conversation).

Slot and Flow Based Conversation

Modern conversational chatbot tools fit into two categories: flow and slot. Flow or script based agents use a flow diagram or script to define the conversation. The agent asks the user questions and based on the user's response, the engine follows the flow diagram or script. The conversation is very tightly controlled requiring only simple NLP.

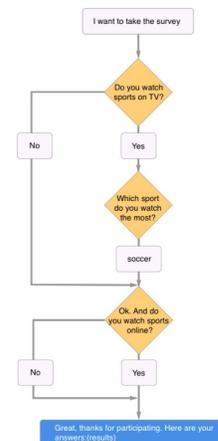


Figure 7. Flow or script based conversation [50]

Flow or script based systems are very simple, but require the creator to anticipate every possible user path, figure 7. If the user responds in an an anticipated way, the agent does not have a path to proceed.

Slot based conversational chatbot tools are smarter requiring more advanced NLP. Slots represent data the slot engine is configured to acquire from the user. The agent creator needs only to define the data they want to acquire, the slot engine guides the conversation with the user to acquire the desired data.

Retrieval-Based vs. Generative Model Responses

“Retrieval-based models (easier) use a repository of predefined responses and some kind of heuristic to pick an appropriate response based on the input and context. The heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of Machine Learning classifiers. These systems don’t generate any new text, they just pick a response from a fixed set.” [49]

“Generative models (harder) don’t rely on pre-defined responses. They generate new responses from scratch. Generative models are typically based on Machine Translation techniques, but instead of translating from one language to another, we “translate” from an input to an output (response).” [49]

Ending the Conversation

The goal of the chatbot is to help the customer. Unfortunately, there will be situations when the chatbot cannot help a customer. To avoid stressing the customer, the chatbot needs to know when to quit and get help from a human. [15,16,17,20,21]

Using sentiment analysis, if the chatbot detects the customer is getting stressed, it should refer the conversation log to a human. [20] If the customer asks a question the chatbot cannot answer, the chatbot should ask for more information from the user. The chatbot uses intent analysis to determine if it is making progress towards finally answer the question. [25] In the event that the chatbot is not decreasing the distance between the customer’s intent and the and the provided intent, the chatbot needs to escalate the conversation log to a human.

When the customer first connects to the chatbot, the chatbot should ask for the customer’s email address.

“Hello, I am a chatbot here to help you. Please enter your email address. If I cannot help you I will send a log a human who will get back to you as soon as possible.”

.... conversation

“My apologies for not being able to answer your question. A log of this conversation has been sent to a human agent. The human agent will get back to you with an answer as soon as possible.”

Chatbot Architecture

Up until very recently, chatbots have only supported a single adjacency pair. This is referred to as a one-shot conversation. Siri, Alexa, and Google assistant are examples of speech based one-shot conversation systems. On the leading edge are chatbots that can support multiple adjacency pairs. Remembering states between pairs and capable of associating data in different adjacency pairs (being able to maintain the conversation).

A chatbot system has four components: front-end, knowledge base, back-end and corpus. The front end is responsible for communicating with the user. It uses natural language processing and AI to determine the user’s intent. It prompts the user when needed for more information. It provides a response to the users intent.

The knowledge base represents the chatbots knowledge in a format consumable by the front end. The knowledge base is created by the back-end.

The back-end consumes the domain’s corpus and creates the knowledge base.

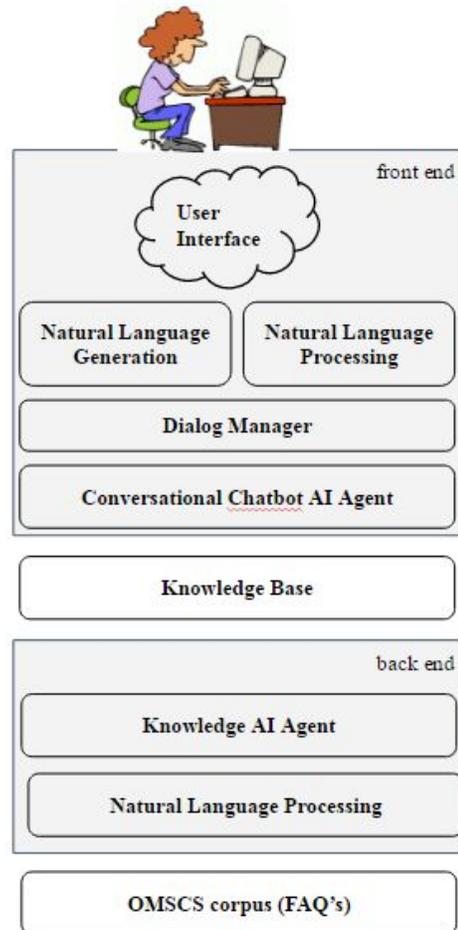


Figure 8. The components of a conversational chatbot.

Figure 8 shows a simplified block diagram of a conversational chatbot. The user interface is used to interact with the AI agent. In most systems the user interface is a web page.

The natural language section is made up of the natural language processing (NLP) block and natural language generation (NLG) block. The NLP block breaks the user's text into intents, entities, and/or actions. This classification is done using either traditional AI techniques or neural networks. NLG is accomplished using templates and a text database stored in the knowledge base.

The dialog manager and conversational chatbot AI (CCAI) manage the conversation. Creating responses based on user input and data from the knowledge base. The CCAI is also responsible for common sense reasoning. Common sense reasoning makes assumptions based on the knowledge corpus.

The knowledge base is the core and personality of the AI agent. The data in the knowledge base is formatted specifically for the chatbot. The goal is to output a response as quickly as possible. The knowledge base takes in entities, intents, and/or actions and outputs either a text response, or a request for more data. The CCAI creates a response based on the knowledge base output. If the output is a text response, the text is forwarded to the user. If the output is a request for more data, the CCAI and dialog manager works with the text generator to generate a response.

The knowledge AI (AIA) agent uses NLP to process the domain corpus in the form of PDF's and web pages in English. The AIA uses NLP to classify the domain corpus text into intents, entities, and/or actions. The domain corpus is classified and tagged in the knowledge base.

Corpus

The corpus is defined by and represents the domain. It can be unstructured or structured, human readable (text) or binary data (possibly compressed). As the name implies, structured data has a format or organization. Most human readable prose have some structure (sentences and paragraphs). Most documents have additional structure in the form of sections and tags in the form of headings. Finally, XML files (and similar formats) are tightly structured with well defined tags.

Frequently Asked Questions (FAQ's)

FAQ's are documents formatted in question/answer pairs. This structure is a natural candidate for a chatbot. A portion of the OMSCS FAQ is shown in appendix A. Documents structured in sentences and paragraphs can be converted into a FAQ format using natural language processing.

Document to FAQ Conversion

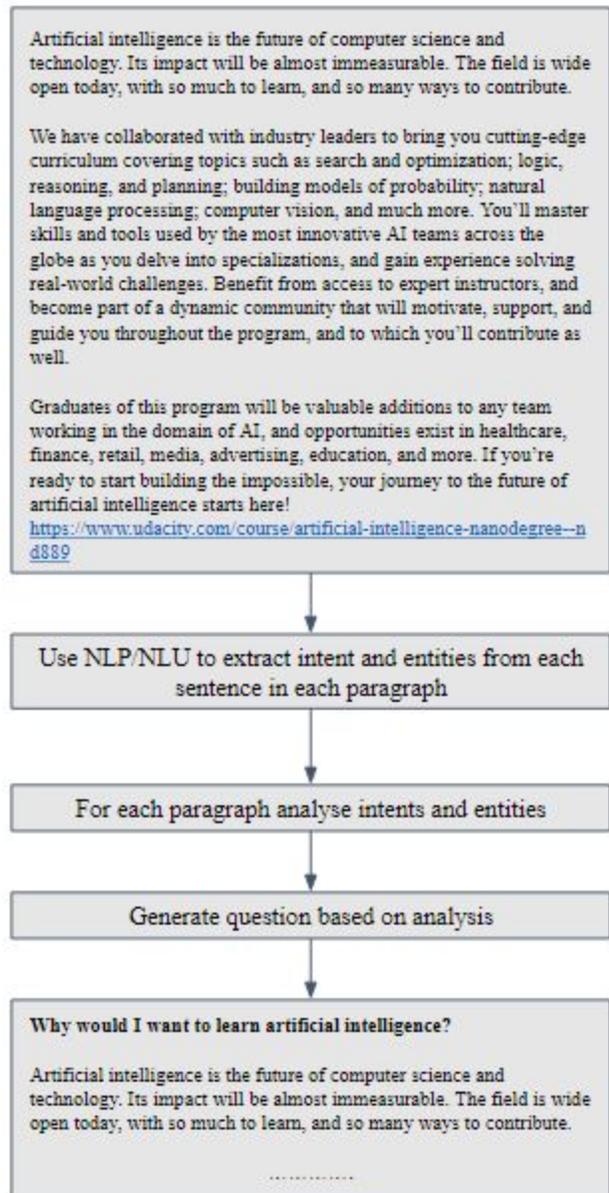


Figure 9. FAQ Generator

Figure 9 is a proposed FAQ generator. The input text is parsed into paragraphs. Each sentence in the paragraph is passed to a Natural Language Understanding (NLU) library. The resulting intents and entities are used to generate a question representing the information in the paragraph. The process repeats for the next paragraph.

Finally, paragraphs with common intents and entities are combined into single questions and the paragraphs are combined to provide a single response.

CHATBOT PLATFORMS AND TOOLS

“Marketing motivations cannot be denied, but if chatbots meet the high expectations of the users, they will become indispensable tools for many use cases. The importance that tech giants like Google, Facebook, Microsoft, IBM and Amazon are giving to chatbots is a strong indicator that this technology will play a key role in the future.” [46]

The last few years have seen an explosion of chatbot tools and platforms. The success of the Apple Siri, Amazon Echo, and Google Home have created a financial incentive to invest in advanced Natural Language Processing and understanding.

NLP/Text Mining Tools	Features
NLTK http://www.nltk.org	Python library tokenize, tag, entities
TextBlob https://textblob.readthedocs.io/en/dev/	Python library tokenize, tag, entities, sentiment
OpenNLP https://opennlp.apache.org/	Java library tokenize, tag, entities
CoreNLP http://nlp.stanford.edu	Library - multiple bindings tokenize, tag, entities
wit.ai appendix C	Web API Advanced NLP including intent extraction
Lexalytics https://www.lexalytics.com/	Web API Advanced NLP including intent extraction
Microsoft Luis https://www.luis.ai appendix C	Web API Advanced NLP including intent extraction
Google Cloud Natural Language API https://cloud.google.com/natural-language/ appendix C	Web SPI tokenize, tag, entities, sentiment
FastText https://github.com/facebookresearch/fastText	Library with multiple bindings for fast text representation and classification - word vectors

Table 1. Natural Language Processing systems

Chatbot Languages	Description
AIML http://www.alicebot.org/aiml.html	“AIML (Artificial Intelligence Markup Language) is an XML-compliant language that's easy to learn, and makes it possible for you to begin customizing an Alicebot or creating one from scratch within minutes.” [35]
ChatScript http://chatscript.sourceforge.net/	“ChatScript is a "next Generation" chatbot engine, based on the one that powered Suzette, that won the 2010 Loebner Competition. ChatScript has many advanced features and capabilities that, when properly utilized, permit extremely clever bots to be programmed. There is also a potentially useful ontology of nouns, verbs, adjectives, and adverbs for understanding meaning.” [36]
RiveScript https://www.rivescript.com/	“RiveScript is a simple scripting language for giving intelligence to chatbots and other conversational entities. It's a plain text, line-based scripting language with goals of being simple to learn, quick to type, and easy to read and maintain.” [53]
Rasa https://rasa.ai/	“Machine Learning based dialogue management and natural language understanding. Rasa is used to build next-generation bots and assistants that engage in human-like, layered conversations.” [54]

Table 2. Chatbot Languages

Chatbot Platforms	Description
Watson https://www.ibm.com/cloud-computing/bluemix/services	Offers various services that can be used for chatbots front and back ends. Behind in conversational chatbot technology.
Amazon LEX https://aws.amazon.com/lex/ appendix C	Amazon Lex plus Lambda provides a chatbot front end with optional speech support via Alexa.
Microsoft Bot https://dev.botframework.com/	front/back end Chatbot front and back end..
Microsoft Azure QnA https://azure.microsoft.com/en-us/services/cognitive-services/ appendix D	Knowledge base builder, back end. Converts FAQ's into knowledge bases.
Google knowledge graph API https://developers.google.com/knowledge-graph/	Knowledge Graph API back end.
agent.ai	A full chatbot service with front and back end services.
Pandorabots https://www.pandorabots.com/	Commercial chatbot based on ALICE and AIML. Front and back end.
api.ai appendix C	Intent parser (NLU) with slot support..
wit.ai appendix	Web interface Advanced NLP including intent extraction

Table 3. Chatbot Platforms

Tables 1-3 show only a small portion of the available chatbot tools. As shown in figure 1, there are many more <https://chatbotsjournal.com/25-chatbot-platforms-a-comparative-table-aeefc932eaff>.

Details about wit.at, api.ai, and Google Natural Language API can be found in appendix C. NLTK, TextBlob, OpenNLP, and CoreNLP are all open source NLP libraries. Lexalytics is a NLU library that provides NLP plus intent extraction and sentiment analysis.

FastText

A metric is required to determine the similarity of two words. A popular metric compares words by generating a vectors for the word. The distance between words is than the Euclidian distance or Cosine similarity between vectors. FastText expands the concept of word vectors by creating vectors for subsets of the word (n-grams). [55]

FastText also supports text classification. Text classification can be used for sentiment analysis or intent detection for a group of utterances. [56]

AIML

Table 2 shows some popular chatbot programming languages. AIML is a XML like chatbot language developed in 1995. AIML is very powerful but the XML dialect is cumbersome. AIML requires an exact match with the user's input, but it supports wildcards (weak pattern matching). [35]

Chatscript

Chatscript is a text based scripting language that came out in 2010. Its pattern matching is significantly better than AIML. It is more difficult to program, but it is much more powerful. At this point in time, ChatScript does not have a lot of support outside its community.

Rasa

“Rasa NLU is an open source tool for intent classification and entity extraction. You can think of it as a set of high level APIs for building your own language parser using existing NLP and ML libraries.” “Rasa NLU is an open source tool for intent classification and entity extraction.” [54] Rasa can be used as an open source replacement for Wit.ai, API.ai, or LUIS.

Wit.ai

Wit.ai, API.ai, LUIS, and Amazon Lex are cloud based chatbot platforms. Wit.ai is supported by Facebook. It supports two modes: NLU and story. NLU mode supports grouping utterances into intents and extracting entities. Story mode supports NLU and flow based conversation. [50] “*Stories are the key concept to model the behavior of a chatbot with Wit.ai. Each story represents an example of a possible conversation. It should be noted that “intent” is no longer a concept but a user entity, non mandatory. This was a change of greater impact in Wit.ai, motivated by the fact that a complex chatbot needs a lot of intents that can, in some way, be grouped in stories. Bot developers basically teach Wit.ai by example. The subjacent idea is that when a user writes “similar” requests, Wit.ai will process the request, extract the entities and apply the logic defined by the developer. A story can be seen as a graph of user intents. You can add branches that are triggered on conditions such as the existence or not of specific variable values, that are extracted from the user input. This allows you to define a conversation flow. Moreover, you have a bookmark mechanism, used to jump between intents and*

also between stories.” [46]

API.ai

API.ai is supported by Google. “*Intents and Contexts are the key concepts to model the behavior of a chatbot with Api.ai. Intents creates links between what a user says and what action should be taken by the bot. Contexts are string values, useful for differentiating requests which might have different meaning depending on previous requests.*

Basically, when Api.ai receives a user request, it is first classified to determine if it matches a known intent. Api.ai proposes a “Default Fallback intent” to deal with requests that do not match any user intent.” “This mechanism of intents and contexts allows to create state machines that model large and complex flows.” [46]

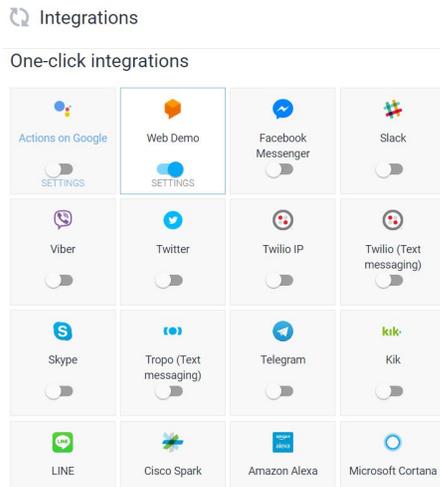


Figure 10. One-Click Integrations

API.ai provides multiple one-click integrations (figure 10). Web demo makes it easy to create a public demo without any server code development. API.ai provides free hosting for testing making it easy to get a chatbot up and running quickly.

API.AI SDK's

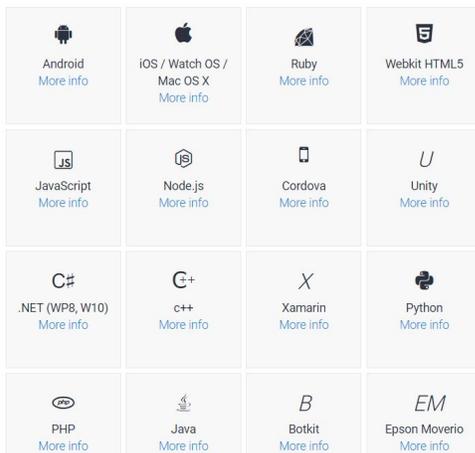


Figure 11. API.ai SDK's

Multiple SDK's are also available for accessing the API.ai chatbot (figure 11) from Python, C, Javascript, Node.JS,

Microsoft LUIS

LUIS is supported by Microsoft. “Utterances are input from the user that your app needs to interpret. To train LUIS to extract intents and entities from them, it's important to capture a variety of different inputs for each intent. Active learning, or the process of continuing to train on new utterances, is essential to machine-learned intelligence that LUIS provides.” [45] LUIS supports SDK's for Android, Node.JS, Python, and Windows.

Amazon Lex

Lex is supported by Amazon. Lex is part of the very successful Amazon Alexa ecosystem. “*Amazon Lex is a service for building conversational interfaces using voice and text. Powered by the same conversational engine as Alexa, Amazon Lex provides high quality speech recognition and language understanding capabilities, enabling addition of sophisticated, natural language ‘chatbots’ to new and existing applications. Amazon Lex reduces multi-platform development effort, allowing you to easily publish your speech or text chatbots to mobile devices and multiple chat services, like Facebook Messenger, Slack, or Twilio SMS. Native interoperability with AWS Lambda, AWS MobileHub and Amazon CloudWatch and easy integration with many other services on the AWS platform including Amazon Cognito, and Amazon DynamoDB makes bot development effortless.” “Amazon Lex leverages AWS Lambda for Intent fulfillment, Amazon Cognito for user authentication, and Amazon Polly for text to speech.” [40]*

Amazon Lex supports SDK's for Java, JavaScript, Python, CLI, .NET, Ruby on Rails, PHP, Go, and CPP. Instructions are provided for integration with Slack, SMS, and Facebook messenger.

When Lex detects an intent, it can either return the information to the client or call a Amazon Lambda function. Lambda enables the execution of code (Node.js, Python, Java, and C#) on a cloud server. When Lex detects an intent, it executes the Lambda function on the cloud and passes the function a json file. The Lambda function returns a json file to Lex telling Lex what to do next.

Amazon Simple Storage Service (S3) is a cloud storage system. Using a simple API encapsulated in the Amazon boto3 Python library, Lambda can access files stored in S3.



Figure 12. Amazon Components to Create Chatbot

TOOL EVALUATION RESULTS

Chatbot conversation is focused on the user’s needs. The quality of the chatbot can be assessed by how efficiently it meets these needs. Has the chatbot provided the information the user is looking for?

Test Procedure

NLU tools: Facebook Wit.ai, Microsoft LUIS, Google API.ai, and Amazon Lex were tested. Each tool was trained with the same utterances from appendix B. Each tool was than passed the same test questions. The test questions included equal utterances along with derivative utterances. The responses from each tool is recorded in table 4.

NLU Test Results

Test Question	WIT	LUIS	API	LEX
How many hours are required to graduate?	84	21	68	NA
How many classes are required to graduate?	56	16	40	NA
When can I apply?	99	100	100	NA
Where can I apply?	99	100	100	NA
What is the application process?		6		NA
What is the admission process?		5		NA
How does the admission process work?		7		NA
Is the GRE required?	99	100	100	NA
What is the minimum GRE score required?	99	100	84	NA
What are foundation courses?	93	100	74	NA
How many foundation courses are required?	55	57	41	NA
What are the admission criteria?	67	99	49	NA
where do I find the admission criteria?		87	35	NA

Table 4. Comparing NLU Tools (front ends)

Note: integers represent percent confidence of intent. The higher the confidence the better the user utterance matched the trained intent. Amazon Lex does not provide a confidence score in its response.

Green indicates the NLU matched the test question to the correct intent. Red indicates a test question/intent match failed.

Wit.ai

Wit.ai response

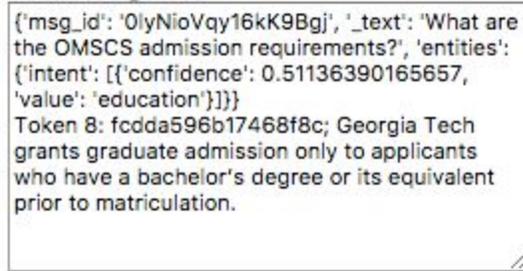


Figure 13. Wit.ai response followed by decoded intent

Wit.ai provides a confidence score in it’s raw response along with the input text and the decoded intent (‘value’). The response shown in figure 13 is an incorrect response to the question, “What are the OMSCS admission requirements?” This question should have been categorized as ‘admission’ as opposed to ‘education’.

```
20 lines (18 sloc) | 788 Bytes
1 from wit import Wit
2
3 class myWit:
4     def __init__(self, jsonData):
5         self.client = Wit(access_token='AA035R25HFJLRCBV5X6A0E7DF5N3TDGN')
6         self.jsonData = jsonData
7
8     def GetIntent(self,msg):
9         resp = self.client.message(msg)
10        retResponse = (str(resp)+'&#13;&#10;')
11        if 'intent' in resp['entities']:
12            intent = resp['entities']['intent'][0]['value']
13            confidence = resp['entities']['intent'][0]['confidence']
14            #
15            #         retResponse = 'category: '+intent+'&#13;&#10;'
16            #         retResponse += 'confidence: '+str(confidence)+'&#13;&#10;'
17            retResponse += self.jsonData[intent]['response']
18        else:
19            retResponse += self.jsonData['None']['response']
20        return retResponse
```

Figure 14. Accessing Wit.ai NLU using the SDK

Wit.ai provides a Python SDK to access the tool. [50] Figure 14 shows the code used by the test tool to access the Wit.ai tool.

When Wit.ai cannot match the input text to an intent, it’s response does not include an intent field. Thus no confidence is provided when a match does not occur.

LUIS

LUIS response

```
{'query': 'What are the OMSCS admission requirements?', 'topScoringIntent': {'intent': 'education', 'score': 0.08426078}, 'intents': [{'intent': 'education', 'score': 0.08426078}, {'intent': 'curriculum', 'score': 0.0575854965}, {'intent': 'None', 'score': 0.03906847}, {'intent': 'recommendation', 'score': 0.028436454}, {'intent': 'admission', 'score': 0.006813628}, {'intent': 'gre', 'score': 6.903379e-05}, {'intent': 'foundation_courses', 'score': 1.47961528e-05}, {'intent': 'application', 'score': 1.01611954e-06}, {'intent': 'transcript', 'score': 7.94046e-07}, {'intent': 'toefl', 'score': 7.136236e-07}], 'entities': []}
Token 8: fcdda596b17468f8c; Georgia Tech grants graduate admission only to applicants who have a bachelor's degree or its equivalent prior to matriculation.
```

Figure 15. LUIS response followed by decoded intent

LUIS returns all intents, each with a confidence score, along with a 'topScoringIntent'. Figure 15 shows the LUIS response to the question, "What are the OMSCS admission requirements?" The top scoring intent is 'education' with a confidence of 8.4%, an incorrect categorization. This question should have been categorized as 'admission' as opposed to 'education'.

20 lines (16 sloc) | 796 Bytes

```
1 import urllib.request
2 import urllib.parse
3 import json
4
5
6 class myLUIS:
7     def __init__(self, jsonData):
8         self.jsonData = jsonData
9         self.url = 'https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/5dd6e
10
11     def GetIntent(self, msg):
12         urlQuestion = urllib.parse.quote(msg)
13         webURL = urllib.request.urlopen(self.url+urlQuestion)
14         resp = webURL.read()
15         encoding = webURL.info().get_content_charset('utf-8')
16         data = json.loads(resp.decode(encoding))
17         intent = data['topScoringIntent']['intent']
18         # confidence = data['topScoringIntent']['score']
19         return str(data) + "&#13;&#10;" + self.jsonData[intent]['response']
```

Figure 16. Accessing LUIS NLU using urllib

Figure 16 shows how LUIS is accessed using the Python urllib library by the test tool. [45] Note: the url is cutoff in figure 16. The full url is available in the Github repository.

API.ai

API.ai response

```
{'id': '2fc7f58c-9424-4f16-a2f8-581d52070c83', 'timestamp': '2017-07-28T01:39:44.831Z', 'lang': 'en', 'result': {'source': 'agent', 'resolvedQuery': 'What are the OMSCS admission requirements?', 'action': '', 'actionIncomplete': False, 'parameters': {}, 'contexts': [], 'metadata': {'intentId': '5c7ada13-bc9a-4706-9985-44e34e925f8a', 'webhookUsed': 'false', 'webhookForSlotFillingUsed': 'false', 'intentName': 'admission'}, 'fulfillment': {'speech': 'Token 2: fcdda596b17462c93 - Preferred qualifications for admitted OMS CS students are an undergraduate degree in computer science or related field (typically mathematics, computer engineering or electrical engineering) from an accredited institution with a cumulative GPA of 3.0 or higher. Applicants who do not meet these criteria will be evaluated on a case-by-case basis; significant professional or other work experience with supporting recommendations may qualify as an adequate substitute for the appropriate academic credentials, however work experience will not take the place of an undergraduate degree. Georgia Tech will not admit applicants into the OMS CS degree program without the minimum qualifications for success. The Georgia Tech minimum criteria used in determining each applicant's eligibility for consideration shall include: Evidence of award of a bachelor's degree or its equivalent (prior to matriculation) from a recognized institution, demonstrated academic excellence, and evidence of preparation in their chosen field sufficient to ensure successful graduate study; and For international applicants, satisfactory scores on the Test of English as a Foreign Language (TOEFL). All applications are reviewed by a faculty committee to ensure that those admitted can succeed in the program. Applicants who are selected for admission will be conditionally admitted into the degree program and must pass their first two OMS CS foundational courses with a grade of B or better to be fully admitted. Students do not need to take these foundational courses before registering for elective courses, however we recommend students take at least one foundational course per semester in which they are enrolled until they are fully admitted to Georgia Tech.'}, 'messages': [{'type': 0, 'speech': 'Token 2: fcdda596b17462c93 - Preferred qualifications for admitted OMS CS students are an undergraduate degree in computer science or related field (typically mathematics, computer engineering or electrical engineering) from an accredited institution with a cumulative GPA of 3.0 or higher. Applicants who do not meet these criteria will be evaluated on a case-by-case basis; significant professional or other work experience with supporting recommendations may qualify as an adequate substitute for the appropriate academic credentials, however work experience will not take the place of an undergraduate degree. Georgia Tech will not admit applicants into the OMS CS degree program without the minimum qualifications for success. The Georgia Tech minimum criteria used in determining each applicant's eligibility for consideration shall include: Evidence of award of a bachelor's degree or its equivalent (prior to matriculation) from a recognized institution, demonstrated academic excellence, and evidence of preparation in their chosen field sufficient to ensure successful graduate study; and For international applicants, satisfactory scores on the Test of English as a Foreign Language (TOEFL). All applications are reviewed by a faculty committee to ensure that those admitted can succeed in the program. Applicants who are selected for admission will be conditionally admitted into the degree program and must pass their first two OMS CS foundational courses with a grade of B or better to be fully admitted. Students do not need to take these foundational courses before registering for elective courses, however we recommend students take at least one foundational course per semester in which they are enrolled until they are fully admitted to Georgia Tech.'}], 'score': 0.610000143051147}, {'source': 'domains', 'resolvedQuery': 'What are the OMSCS admission requirements?', 'actionIncomplete': False, 'metadata': {}, 'fulfillment': {}, 'score': 0.0}, {'status': {'code': 200, 'errorType': 'success'}, 'sessionId': 'b14b22eb-3914-4031-be5e-a7f37e8e5f24'}
```

Figure 17. API.ai raw response

API.ai support assigning responses to intents. As shown in figure 17, API.ai returns the intent, score, and a response. Intent decoding is not required. Figure 17 shows the API.ai response to the question, "What are the OMSCS admission requirements?" The returned intent 'admission' with a confidence score of 61% is correct. The 'context' field supports the concept of a thread connecting questions and answers in a conversation; a dialog.

26 lines (24 sloc) | 963 Bytes

```
1 import uuid
2 import apiai
3 import json
4
5 class myAPI:
6     def __init__(self, jsonData):
7         ai = apiai.ApiAI('390bfd55dc8c4a979998b32b8e884c64')
8         self.request = ai.text_request()
9         self.request.session_id = str(uuid.uuid4())
10        self.jsonData = jsonData
11
12    def GetIntent(self, msg):
13        self.request.query = msg
14        resp = self.request.get_response()
15        data = json.loads(resp.read())
16        retResponse = (str(data) + "&#13;&#10;")
17        # if 'intent' in resp['entities']:
18        #     intent = resp['entities']['intent'][0]['value']
19        #     confidence = resp['entities']['intent'][0]['confidence']
20        #     if 0:
21        #         retResponse = 'category: '+intent+'&#13;&#10;'
22        #         retResponse += 'confidence: '+str(confidence)+'&#13;&#10;'
23        #         retResponse += self.jsonData[intent]['response']
24        #     else:
25        #         retResponse += self.jsonData['None']['response']
26        return retResponse
```

Figure 18. Accessing API.ai NLU using the SDK

API.ai is accessed using the apiai Python SDK. [44] For the

test tool, the raw response from API.ai was output to the text box. No intent decoding is performed by the code shown in figure 18.

Amazon Lex

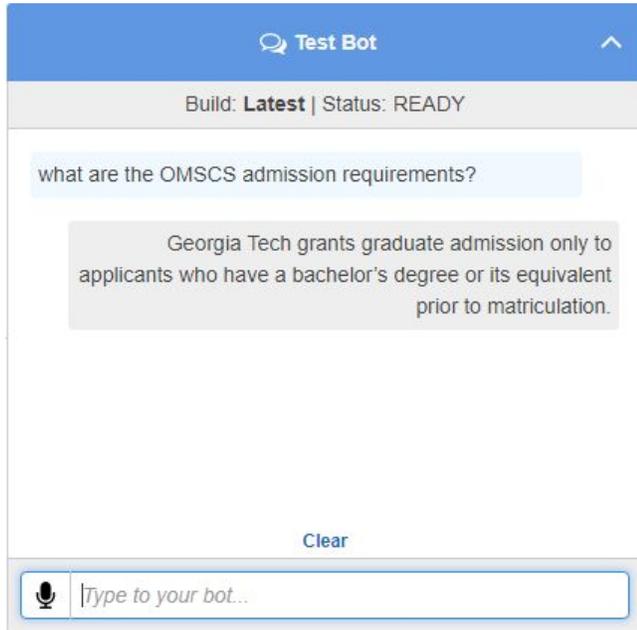


Figure 19. Amazon Lex Editor Test Tool

The test tool’s execution engine (Pythonanywhere.com) does not have the Amazon Lex server url whitelisted. As a result, the Amazon Lex server could not be accessed from Pythonanywhere (free version). Amazon Lex can be accessed from a Pythonanywhere.com paying account.

The Amazon Lex editor ‘Test Bot’ tool (shown in figure 19) was used to test the Amazon Lex NLU. When asked, “What are the OMSCS admission requirements?” Lex incorrectly replies with the ‘education’ intent.

```

03:27:10 [DEBUG] 2017-07-28T03:27:10.612Z a41306c3-7344-11e7-ad0e-3da622a3c740 [u'currentIn
[DEBUG] 2017-07-28T03:27:10.612Z a41306c3-7344-11e7-ad0e-3da622a3c740 {u'currentIntent': {u'slots': {}, u'name':
u'education', u'confirmationStatus': u'None'}, u'bot': {u'alias': None, u'version': u'$LATEST', u'name': u'omscs'},
u'userId': u'j9n772i30rx6pagwiw8az0g5hcucoq7', u'inputTranscript': u'what are the OMSCS admission requirements?',
u'invocationSource': u'FulfillmentCodeHook', u'outputDialogMode': u'Text', u'messageVersion': u'1.0',
u'sessionAttributes': {}}
```

Figure 20. Amazon Lex Raw Response

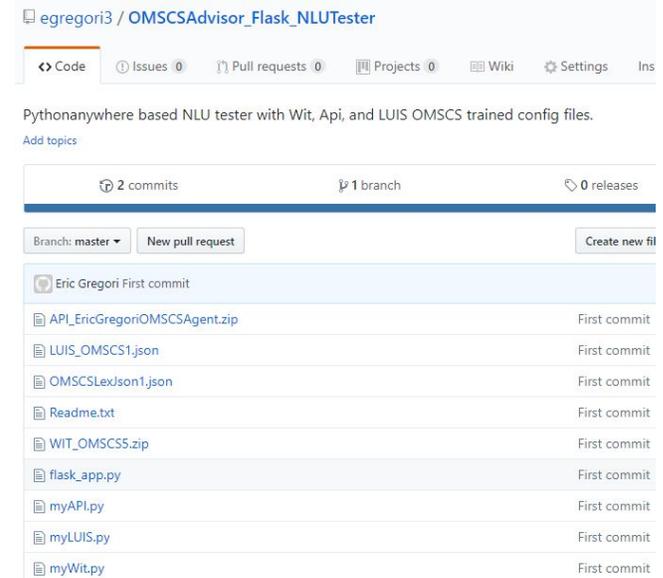
Figure 20 is the raw Amazon Lex response from figure 19. This data was acquired using Amazon Lambda and CloudWatch.

```

{u'currentIntent': {u'slots': {}, u'name': u'education',
u'confirmationStatus': u'None'}, u'bot': {u'alias': None,
u'version': u'$LATEST', u'name': u'omscs'}, u'userId':
u'j9n772i30rx6pagwiw8az0g5hcucoq7',
u'inputTranscript': u'what are the OMSCS admission
requirements?',
u'invocationSource':
u'FulfillmentCodeHook', u'outputDialogMode': u'Text',
u'messageVersion': u'1.0', u'sessionAttributes': {}}
```

Test Tool Source Code and NLU Export Files

Each tool was configured using its web editor as shown in appendix E. The data from each tool was exported using its export function. These export files, along with the Pythonanywhere code discussed in this section is available from Github:



https://github.com/egregori3/OMSCSAdvisor_Flask_NLUTester

OMSCS ADVISOR CHATBOT

Have a beginning, a middle and an end: A purpose-driven chatbot will guide a user through to an intended end goal (conduct a transaction, serve a piece of content). Consider the journey a user will take when they encounter the bot, how the customer will move into an interaction and out of it. Off-script conversations are also inevitable, and good chatbot designs build in paths to navigate such conversations back on-track.” [11]

Block Diagram

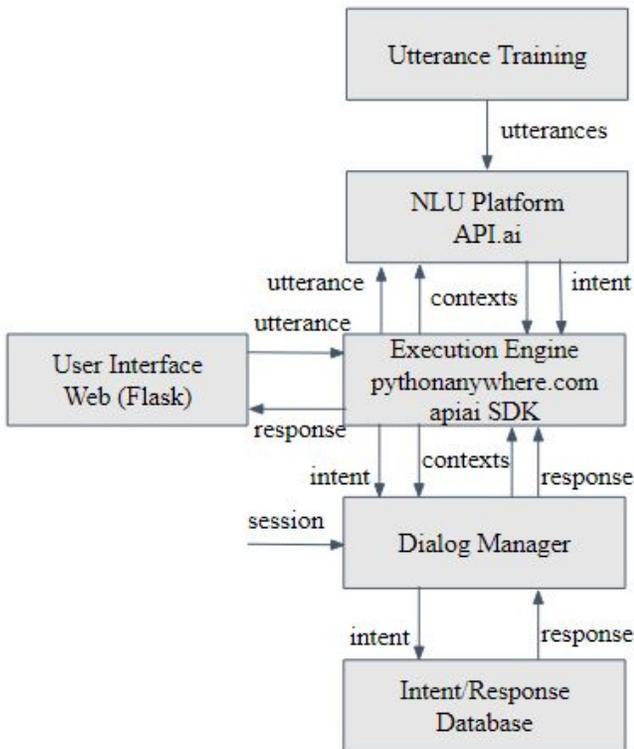


Figure 21. Block Diagram of OMSCS Chatbot

Execution Engine/User Interface

The execution engine runs on Pythonanywhere.com and is written in Python. It manages the session and moves information between the NLU platform, dialog manager (DM), and the user interface. The user interface is a web page served by the Python Flask library. The user interface is shown in appendix F.

NLU Platform

Google API.ai was chosen as the NLU platform due to its performance in the NLU test (table 4). LUIS was also considered and will be researched later. Amazon Lex has many benefits include Amazon Lambda and S3 but it currently does not return a confidence score. Facebooks Wit.ai is similar to API.ai in features and accuracy. API.ai was chosen over Wit.ai because of a very powerful feature

called ‘contexts’.

```
{
  "id": "26aacad9-3f79-48f8-bf8e-ff0768e3c51c", "timestamp": "2017-09-20T15:00:00.000Z",
  "lang": "en",
  "result": {
    "source": "agent",
    "resolvedQuery": "this is a test?",
    "action": "input.unknown",
    "actionIncomplete": "False",
    "parameters": {},
    "contexts": [],
    "metadata": {
      "intentId": "9866c10a-09fd-4ae0-9672-fe8facd84f4c",
      "webhookUsed": "false",
      "webhookForSlotFillingUsed": "false",
      "intentName": "Default Fallback Intent"
    },
    "fulfillment": {
      "speech": "I can only answer questions about OMSCS a",
      "messages": [
        {
          "type": 0,
          "speech": "I can only answer questions about O"
        }
      ],
      "score": 1.0
    },
    "status": {
      "code": 200,
      "errorType": "success"
    },
    "sessionId": "a125b95a-92f8-4ef3-aa7b-af6920201a7d"
  }
}
```

Figure 22. Example API.ai response (Json)

Figure 22 shows an example response from the API.ai tool. Appendix F contains a complete description of all the fields. Some fields useful for a conversational agent include: ‘contexts’, ‘metadata’, ‘score’, and ‘sessionId’. ‘Metadata’ contains the results of the query classification. ‘SessionId’ is created by the client and passed to API.ai in a query. The same session ID is returned in the response. SessionID can be used to support multiple sessions. [44]

API.ai ‘contexts’

API.ai ‘contexts’ are named binary variables for managing state in a dialog. Intents can be enabled or disabled by contexts. Intents can set contexts but not clear them. If an intent is associated with a context, it will only be returned if the context is true.

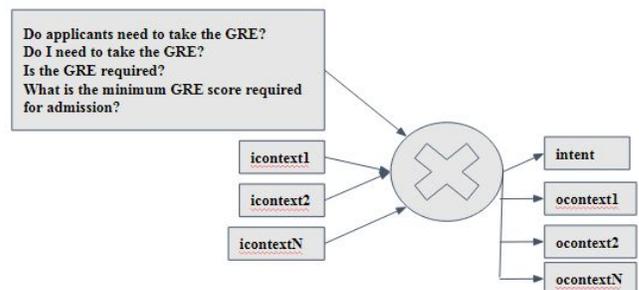


Figure 23. Input ‘contexts’ are ‘anded’ together

As shown in figure 23, the input ‘contexts’ are anded together. They must all be set in order for the intent to fire. When the intent fires, it sets output ‘contexts’.

‘Contexts’ have a lifetime. Each time a ‘context’ is matched its lifetime counter decrements. When the ‘context’ reaches 0 the context is disabled.

Utterance Training

API.ai classifies utterances into intents. The classifier is trained with multiple utterances of different wording. Initial training was done from appendix B. After initial training the chatbot was exposed to students in the Summer CS6460 Education Technology class. After 10 days the chatbot recorded approximately 70 sessions and 330 queries.

The API.ai tool logs each session and the requests (user utterances) received during the session. With each utterance it logs its categorization of the utterance (intent).

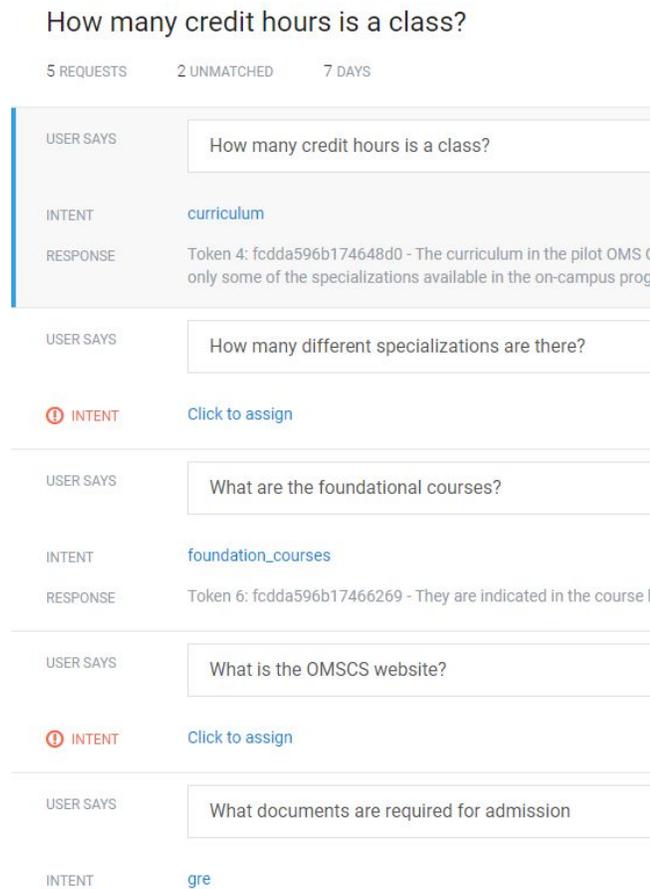


Figure 24. Training API.ai on new utterances from users

Figure 24 shows an example session log entry with 5 requests. For each utterance, the chatbot programmer tags it as correctly or incorrectly classified. If the utterance does not fit any current intent, a new intent can be created. After

tagging the API.ai tool incorporates the new utterances into its classification model.

Dialog Manager

The goal is to provide the user with the information they are looking for. The first step to achieving this goal is for the user to ask a question. The chatbot attempts to match the question to an intent. If there is a match, the chatbot responds with an answer.

User feedback

At this point, the chatbot does not know if the goal of providing the user the information they are looking for has been achieved. To get this feedback, the chatbot asks the user, “Is this the answer you are looking for?” If the user answers in the affirmative, the chatbot has achieved its goal.

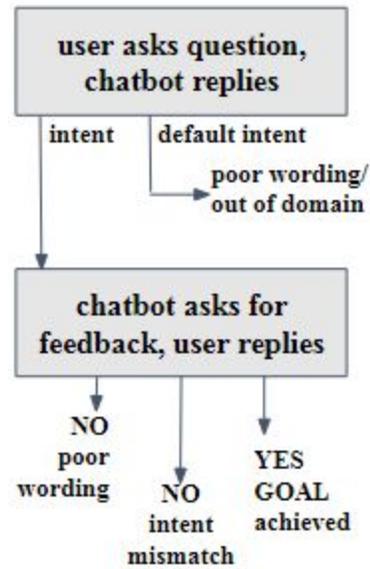


Figure 25. A Dialog Unit (DU)

Negative user feedback indicates the user has not received the information they are looking for. This can be caused by, a poorly worded question or an incorrect utterance/intent match (figure 25).

Poor wording/Intent mismatch

For poor wording or intent mismatch, additional natural language processing is required. The entities from the question can be used to guide the user towards a better question.

Future revisions of the OMSCS advisor tool will experiment with providing better user assistance in question rewording using NLU. For example, the chatbot could suggest questions to the user.

Out of domain question

Default Fallback Intent

1 Fallback intents are triggered if a user's input is not matched by any of the regular intents or enabled domains. [Read more in do](#)

Contexts

Add input context

5 NotAnswered Add output context

Events

Action

input.unknown

Response

DEFAULT +

Text response

- 1 I can only answer questions about OMSCS admissions and curriculum at this time.
- 2 I did not understand your question. I can only answer questions about OMSCS admissions and curriculum at this time.
- 3 I have not been trained on that information. I can only answer questions about OMSCS admissions and curriculum at this time.
- 4 I think the question is outside my domain. I can only answer questions about OMSCS admissions and curriculum at this time.
- 5 Enter a text response variant

Figure 26. API.ai default fallback intent

API.ai provides a default intent that can be used to the domain question (figure 26). Default intents are assumed to be out of domain utterances and provide a response explaining the chatbots domain.

Conversation

A conversation consists of multiple DU's (figure 25). The goal of the conversation is to get an affirmative feedback reply from the user indicating the user has the information they were looking for. Currently the chatbot can only ask the user to reword the question while providing guidance on the supported domain. With additional NLU, the chatbot will be able to guide the user more intelligently.

Research indicates users get frustrated if they do not get their answer after asking the question four times (reworded). To help the user, the chatbot sends the conversation log to a human. The user is told a human will answer their question within 24/48 hours. [21]

Contexts

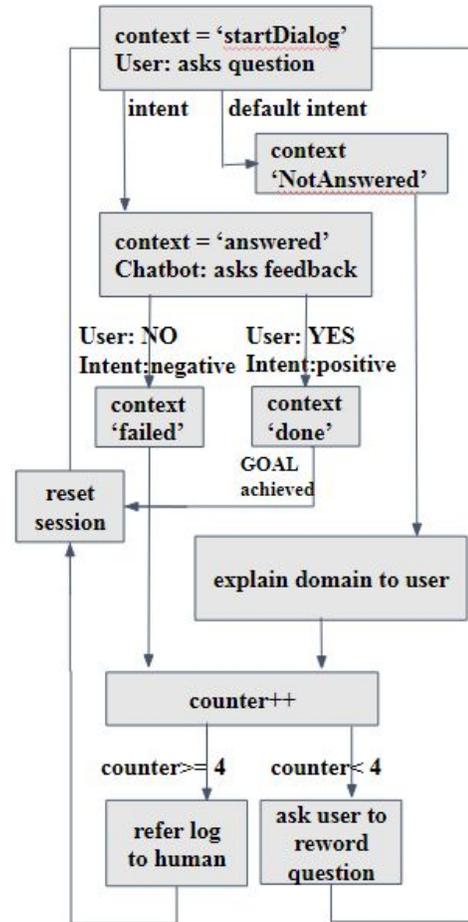


Figure 27. Using 'contexts' to manage dialog

As shown in figure 27, API.ai 'contexts' are used to manage the dialog. The execution engine watches the 'contexts' and executes the flow.

RESULTS

Screenshots of the OMSCS advisor are shown in appendix F. The initial evaluation of the tool was done using the API.ai user interface. The evaluation was well received. The new utterances were used to retrain the API.ai tool.

The current dialog manager is very simple. A more advanced dialog manager is planned for future revisions of the OMSCS advisor tool.

Finally, the current user interface is very limited. It only supports a single session, locking out other users while a session is in progress. The user interface is OK for testing but will need to be replaced for full deployment.

REFERENCES

1. Shaikh, A., Phalke, G., Patil, P., Bhosale, S., & Raghatwan, J. (2016). *A Survey On Chatbot Conversational Systems*. International Journal of Engineering Science, 3117.
2. Kane, D. A. (2016). *The Role of Chatbots in Teaching and Learning*.
3. McTear, M., Callejas, Z., & Griol, D. (2016). *The conversational interface*. Berlin: Springer.
4. Luger, E., & Sellen, A. (2016, May). *Like Having a Really Bad PA: The Gulf between User Expectation and Experience of Conversational Agents*. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 5286-5297). ACM.
5. Susan B.. (1990). *Conversation as direct manipulation: An iconoclastic view*. In The Art of Human-Computer Interface Design. B.K. Laurel (Ed.), Reading, MA: Addison-Wesley
6. <http://www.omscs.gatech.edu/>
7. <https://www.omscs.gatech.edu/prospective-students/numbers>
8. OMS CS Advising Team. (2017). *OMS CS SUMMER 2017 ORIENTATION INFORMATION*
9. <https://qnamaker.ai>
10. <https://aws.amazon.com/lex/>
11. Yee, Y. (2017). *Why is my chatbot not chatting? Demystifying the chatbot hype*. <http://www.campaignasia.com/article/why-is-my-chatbot-not-chatting/436747>
12. Morgan, B. (2017). *How Chatbots Improve Customer Experience In Every Industry: An Infograph*. <https://www.forbes.com/sites/blakemorgan/2017/06/08/how-chatbots-improve-customer-experience-in-every-industry-an-infograph>
13. Patal, S. (2017). *6 Effective Uses for Chatbots in Marketing*. <https://www.forbes.com/sites/sujanpatel/2017/01/21/6-effective-uses-for-chatbots-in-marketing/#1b2c9a973f48>
14. Chorny, V. (2017). *5 industries that benefit from chatbots already*. <https://chatbotnewsdaily.com/5-industries-that-benefit-from-chatbots-already-b1cee62e7dad>
15. Alzahrani, H. (2016). *Artificial Intelligence and Customer Communication*. Global Journal of Computer Science and Technology, 16(1).
16. Drexler, K. E. (1992). *Nanosystems: Molecular Machinery, Manufacturing, and Computation*. New York: Wiley.
17. Miner, A., Chow, A., Adler, S., Zaitsev, I., Tero, P., Darcy, A., & Paepcke, A. (2016, October). *Conversational Agents and Mental Health: Theory-Informed Assessment of Language and Affect*. In Proceedings of the Fourth International Conference on Human Agent Interaction (pp. 123-130). ACM.
18. BenMark, G., Venkatachari, D. (2016). *Messaging Apps Are Changing How Companies Talk with Customers*. Harvard Business Review. <https://hbr.org/2016/09/messaging-apps-are-changing-how-companies-talk-with-customers>
19. Wellers, D., Elliott T., Noga, M. (2017). *8 Ways Machine Learning Is Improving Companies' Work Processes*. Harvard Business Review. <https://hbr.org/2017/05/8-ways-machine-learning-is-improving-companies-work-processes>
20. High, R. (2017). *Chatbots evolve from greeting tool to customer service must-have*. <https://venturebeat.com/2017/04/18/chatbots-evolve-from-greeting-tool-to-customer-service-must-have/>
21. Accenture Interactive. (2016). *Chatbots in Customer Service*. https://www.accenture.com/t00010101T000000w/_br-pt/_acnmedia/PDF-45/Accenture-Chatbots-Customer-Service.pdf
22. Kane, Danielle A.(2016). *The Role of Chatbots in Teaching and Learning*. In *E-Learning and the Academic Library: Essays on Innovative Initiatives* Location: McFarland. UC Irvine: UC Irvine Libraries. Retrieved from: <http://escholarship.org/uc/item/1hs0k71b>
23. Kerlyl, A., Hall, P., & Bull, S. (2007). *Bringing Chatbots into Education: Towards Natural Language Negotiation of Open Learner Models*. In Applications and Innovations in Intelligent Systems XIV (pp. 179-192). Springer London.
24. Heller, B., Procter, M., Mah, D., Jewell, L., & Cheung, B. (2005, June). *Freudbot: An investigation of chatbot technology in distance education*. In Proceedings of the World Conference on Multimedia, Hypermedia and Telecommunications (pp. 3913-3918).
25. Beaver, I., & Freeman, C. (2016, March).

- Prioritization of Risky Chats for Intent Classifier Improvement.* In The Twenty-Ninth International Flairs Conference.
26. <http://blog.admithub.com/engaging-students-with-chatbots-a-rising-tide>
 27. <http://blog.admithub.com/case-study-how-admithub-is-freezing-summer-melt-at-georgia-state-university>
 28. Page, L. C., & Gehlbach, H. (2017). How an Artificially Intelligent Virtual Assistant Helps Students Navigate the Road to College. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2940297
 29. André, E., T. Rist, and J. Müller. (1997). *WebPersona: A Life-Like Presentation Agent for Educational Applications on the World-Wide Web.* in Workshop on Intelligent Educational Systems on the World Wide Web at Artificial Intelligence in Education. Kobe, Japan.
 30. Shaw, E., W.L. Johnson, and R. Ganeshan. (1999). *Pedagogical Agents on the Web.* in International Conference on Autonomous Agents. ACM Press: Seattle, WA, USA. p. 283-290.
 31. Graesser, A.C., K. Moreno, J. Martineau, A. Adcock, A. Olney, and N. Person. (2003). *AutoTutor improves deep learning of computer literacy: Is it the dialogue or the talking head?* in Artificial Intelligence in Education. Amsterdam: IOS Press.
 32. Lester, J.C., B.A. Stone, and G.D. Stelling. (1999). *Lifelike Pedagogical Agents for Mixed-Initiative Problem Solving.* in Constructivist Learning Environments. User Modeling and User-Adapted Interaction. 9: p. 1-44.
 33. Person, N.K. and A.C. Graesser. (2006). *Pedagogical Agents and Tutors.* in Encyclopedia of Education, J.W. Guthrie, Editor. Macmillan: New York. p. 1169-1172.
 34. Mäurer, D. (2016). *Rational Decision Support with a Natural Language Dialogue System* (Doctoral dissertation, Technische Universität Darmstadt).
 35. <http://www.alicebot.org/about.html>
 36. <http://chatscript.sourceforge.net/>
 37. <https://docs.microsoft.com/en-us/bot-framework/azure-bot-service-quickstart>
 38. <http://docs.pandorabots.com/tutorials/getting-started/>
 39. <https://www.ibm.com/watson/>
 40. <https://aws.amazon.com/lex/>
 41. <https://azure.microsoft.com/en-us/services/cognitive-services/qna-maker/faq/>
 42. <https://cloud.google.com/natural-language/>
 43. <https://googlecloudplatform.github.io/google-cloud-python/stable/language-usage.html>
 44. <https://api.ai>
 45. <https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/Home>
 46. Couto, J. (2017). *Building a Chatbot: analysis & limitations of modern platforms.* <https://tryolabs.com/blog/2017/01/25/building-a-chatbot-analysis--limitations-of-modern-platforms/>
 47. Singh, B. (2017). *Chat Bots — Designing Intents and Entities for your NLP Models.* <https://blogs.msdn.microsoft.com/brijrajsingh/2017/01/29/chat-bots%E2%80%8A-%E2%80%8Adesigning-intents-and-entities-for-your-nlp-models/>
 48. Cambria, E., White, B. (2014). *Jumping NLP Curves: A Review of Natural Language Processing Research.* in MAY 2014 | IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE. <http://www.sentic.net/jumping-nlp-curves.pdf>
 49. Britz, D. (2016). Deep Learning for Chatbots. <http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/>
 50. <https://wit.ai>
 51. Bashmakov, P. (2016). *Advanced Natural Language Processing Tools for Bot Makers – LUIS, Wit.ai, Api.ai and others (UPDATED).* <https://stanfy.com/blog/advanced-natural-language-processing-tools-for-bot-makers/>
 52. Jurafsky, D., Martin, J. H. (2017). *Speech and Language Processing* (3rd ed. draft). Read on July 2017 from: <https://web.stanford.edu/~jurafsky/slp3/>
 53. <https://www.rivescript.com/>
 54. <https://rasa.ai/>
 55. P. Bojanowski*, E. Grave*, A. Joulin, T. Mikolov, *Enriching Word Vectors with Subword Information*
 56. A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, *Bag of Tricks for Efficient Text Classification*

What are the admission criteria for the OMS CS degree program?

Preferred qualifications for admitted OMS CS students are an undergraduate degree in computer science or related field (typically mathematics, computer engineering or electrical engineering) from an accredited institution with a cumulative GPA of 3.0 or higher. Applicants who do not meet these criteria will be evaluated on a case-by-case basis; significant professional or other work experience with supporting recommendations may qualify as an adequate substitute for the appropriate academic credentials, however work experience will not take the place of an undergraduate degree. Georgia Tech will not admit applicants into the OMS CS degree program without the minimum qualifications for success. The Georgia Tech minimum criteria used in determining each applicant's eligibility for consideration shall include:

Evidence of award of a bachelor's degree or its equivalent (prior to matriculation) from a recognized institution, demonstrated academic excellence, and evidence of preparation in their chosen field sufficient to ensure successful graduate study; and

For international applicants, satisfactory scores on the Test of English as a Foreign Language (TOEFL).

I don't have the academic credentials for full admission now, but I would like to work toward that. What advice do you have?

The best advice is to take (and pass) courses in computer science from an accredited institution. If you don't have a computing background, passing these courses will present its own challenges and give you a preview of what the OMS CS curriculum will be like. If you have a computing background (either academically or professionally) and just need to plug a few holes, such courses are a great way to do it. All applicants to the OMS CS or other graduate degree program must meet the Georgia Tech minimum admission criteria (see above), however admission is not guaranteed for those who meet the minimum criteria.

If I meet the admission criteria, am I automatically admitted?

No applicant is automatically admitted. All applications are reviewed by a faculty committee to ensure that those admitted can succeed in the program. Applicants who are selected for admission will be conditionally admitted into the degree program and must pass their first two OMS CS foundational courses with a grade of B or better to be fully admitted. Students do not need to take these foundational courses before registering for elective courses, however we recommend students take at least one foundational course per semester in which they are enrolled until they are fully admitted to Georgia Tech.

What are the foundational courses?

They are indicated in the course listing on the [Program Information](#) page of this website.

What kind of proof of U.S. citizenship is required?

For more information on accepted proof of citizenship, please visit [the Georgia Tech graduate admissions website](#).

I'm not a U.S. citizen and/or don't live in the United States. May I still apply?

Yes, subject to [U.S. export control policy](#).

How does the admissions process work?

Prospective students will have to furnish materials commonly required for graduate admissions (prior degrees, transcripts, etc.). For full information about application requirements, please visit the [Program Information](#) page of this website.

Do applicants need to take the GRE?

No.

When can I apply?

Application periods, including deadlines, will be announced on the OMSCS website under the [Program Information](#) page. You can also receive application information by joining the [OMS CS mailing list](#).

How do I apply?

First, review the information on the [Program Information](#) page, which explains the relevant dates, costs and student information necessary for application. During application windows, a link to the OMS CS application site will be posted on the Program Information page.

If my country's primary language is English, do I still have to provide a TOEFL score?

Exceptions are given to the applicants from countries where English is the SOLE OFFICIAL language of instruction (Australia, Bahamas, Barbados, Canada-except Québec province, England, Ghana, Ireland, Jamaica, Kenya, New Zealand,

Nigeria, Scotland, St. Vincent and the Grenadines, Singapore, Trinidad, Tobago, Uganda and Wales). An applicant is exempt from TOEFL only if they are a naturalized citizen, a Green Card holder or have spent at least one year in residence and enrolled at a U.S. college or university.

Please visit the [Georgia Tech Catalog - TOEFL for International Students page](#) for further information regarding TOEFL requirements.

Can I submit my TOEFL score after the application process ends?

No. TOEFL scores are due upon completion of your application.

[Lawful Presence](#)

Under Policy 4.1.6, the lawful presence of any student applying to an institution must be verified before a final offer of admission can be extended. This policy applies to any student admitted for the Georgia Tech Fall 2011 semester or any semester thereafter. If the student will remain living outside of the U.S. and will be enrolling only in distance education courses, then it is not necessary to verify lawful presence. International students residing outside the United States do not need to submit proof of lawful presence. All other applicants must submit proof of lawful presence. Please visit the [Lawful Presence page](#) of our website for a complete list of acceptable lawful presence documents.

What is meant by official or unofficial transcripts? When can I send them?

Official transcripts can be delivered in several ways. Most common are those delivered directly to Georgia Tech from your academic institution. Some institutions also provide certified PDFs to alumni for use in graduate applications, as well as sealed, physical transcripts. If you are accepted to OMS CS, you will be asked to verify your unofficial transcript provided at the time of application with an official copy through one of these methods.

I am an international student. Do I need to have my transcripts translated or converted?

You will be asked to provide an unofficial English translation of your transcript AND native language transcripts upon application. If your undergraduate institution does not grade on a 4.0 scale, you do NOT need to convert your GPA. The Office of Graduate Studies does not accept self-converted GPA's and will do their own conversion.

How many recommenders will I need? What is the process?

Applicants must procure recommendations from three individuals. As part of the application, applicants will provide their recommenders' email addresses. Those individuals will be contacted through email and provided instructions on how to submit their recommendations online. A physical letter will not be required.

How do I provide criminal and academic misconduct history?

On the OMS CS application, you will be asked to report your own history to be verified by Georgia Tech's Office of Graduate Admissions.

Is there a cap on admissions?

There is no hard cap on admissions, however Georgia Tech and Udacity are controlling enrollments until the necessary infrastructure is in place to deliver a high-quality, supported degree program to enrolled students. The inaugural admissions class for Spring 2014 included about 400 students, and this number will scale up gradually over time.

When will the program start and run until? What's the academic calendar and schedule for the program?

All relevant dates are maintained on the [Program Information](#) page.

What is the curriculum, and how does it compare to the curriculum for the residential degree?

The curriculum in the pilot OMS CS will represent a subset of the on-campus curriculum, allowing for a full MS in computer science but with only some of the specializations available in the on-campus program. The OMS CS curriculum will expand as more courses come online.

I don't have a formal education. Do I still qualify for the program?

Georgia Tech grants graduate admission only to applicants who have a bachelor's degree or its equivalent prior to matriculation.

May I take OMS courses for college credit without being enrolled in the degree program?

No, Georgia Tech does not currently offer OMS courses for credit to students not enrolled in the degree program.

I'm not interested in the degree program, credentials or credit, but I would like to take the courses for free. Is this possible?

Yes, the OMS courses are available for free on the [Udacity website](#). These free courses do not carry college credit and offer

limited academic support.

Can this degree program be audited if I already have an MS CS degree?

Not at first. We will provide additional program tracks following successful launch of the degree program, including possibly an audit track.

Will the degree I receive from the OMS CS program be the same as the on-campus MS in Computer Science or will my degree say "Online"?

Your diploma will read "Master of Science in Computer Science," exactly the same as those of on-campus graduates. There will be no "online" designation for the degrees of OMS CS graduates.

APPENDIX B - EXAMPLE OMSCS KNOWLEDGE BASE DERIVED FROM APPENDIX A

Utterance	Intent/ Entity	Response
<p>Do applicants need to take the GRE? Do I need to take the GRE? Is the GRE required? What is the minimum GRE score required for admission?</p>	‘gre’	Applicants do not need to take the GRE. A GRE score is not required for admission.
<p>What are the minimum education requirements? What level of formal education is required?</p>	‘education’	Georgia Tech grants graduate admission only to applicants who have a bachelor’s degree or its equivalent prior to matriculation.
<p>How many recommenders will I need? Do I need recommendations?</p>	‘recommendat ion’	Applicants must procure recommendations from three individuals. As part of the application, applicants will provide their recommenders’ email addresses. Those individuals will be contacted through email and provided instructions on how to submit their recommendations online. A physical letter will not be required.
<p>Do I need to have my transcripts translated or converted? What is meant by official or unofficial transcripts?</p>	‘transcript’	<p>You will be asked to provide an unofficial English translation of your transcript AND native language transcripts upon application. If your undergraduate institution does not grade on a 4.0 scale, you do NOT need to convert your GPA. The Office of Graduate Studies does not accept self-converted GPA’s and will do their own conversion.</p> <p>Official transcripts can be delivered in several ways. Most common are those delivered directly to Georgia Tech from your academic institution. Some institutions also provide certified PDFs to alumni for use in graduate applications, as well as sealed, physical transcripts. If you are accepted to OMS CS, you will be asked to verify your unofficial transcript provided at the time of application with an official copy through one of these methods.</p>
<p>If my country’s primary language is English, do I still have to provide a TOEFL score?</p>	‘TOEFL’	<p>Exceptions are given to the applicants from countries where English is the SOLE OFFICIAL language of instruction (Australia, Bahamas, Barbados, Canada-except Québec province, England, Ghana, Ireland, Jamaica, Kenya, New Zealand, Nigeria, Scotland, St. Vincent and the Grenadines, Singapore, Trinidad, Tobago, Uganda and Wales). An applicant is exempt from TOEFL only if they are a naturalized citizen, a Green Card holder or have spent at least one year in residence and enrolled at a U.S. college or university.</p> <p>TOEFL scores are due upon completion of your application.</p> <p>Please visit the Georgia Tech Catalog - TOEFL for International Students page for further information regarding TOEFL requirements.</p>
<p>What is the curriculum, and how does it compare to the curriculum for the residential degree? How many total hours are required?</p>	‘curriculum’	The curriculum in the pilot OMS CS will represent a subset of the on-campus curriculum, allowing for a full MS in computer science but with only some of the specializations available in the on-campus program. The OMS CS curriculum will expand as more courses come online.
<p>What are the admission criteria for the OMS CS degree program?</p>	‘admission’	Preferred qualifications for admitted OMS CS students are an undergraduate degree in computer science or related field (typically mathematics, computer engineering or electrical engineering) from an accredited institution with a cumulative GPA of 3.0 or higher. Applicants who do not meet these criteria will be evaluated on a case-by-case basis; significant professional or other work experience with supporting recommendations may qualify as an adequate substitute for the appropriate academic

		<p>credentials, however work experience will not take the place of an undergraduate degree. Georgia Tech will not admit applicants into the OMS CS degree program without the minimum qualifications for success. The Georgia Tech minimum criteria used in determining each applicant's eligibility for consideration shall include:</p> <p>Evidence of award of a bachelor's degree or its equivalent (prior to matriculation) from a recognized institution, demonstrated academic excellence, and evidence of preparation in their chosen field sufficient to ensure successful graduate study; and</p> <p>For international applicants, satisfactory scores on the Test of English as a Foreign Language (TOEFL).</p> <p>All applications are reviewed by a faculty committee to ensure that those admitted can succeed in the program. Applicants who are selected for admission will be conditionally admitted into the degree program and must pass their first two OMS CS foundational courses with a grade of B or better to be fully admitted. Students do not need to take these foundational courses before registering for elective courses, however we recommend students take at least one foundational course per semester in which they are enrolled until they are fully admitted to Georgia Tech.</p>
What are the foundational courses?	'foundation courses'	They are indicated in the course listing on the Program Information page of this website.
Where do I apply? When can I apply?	'application'	<p>Application periods, including deadlines, will be announced on the OMSCS website under the Program Information page. You can also receive application information by joining the OMS CS mailing list.</p> <p>The Program Information page explains the relevant dates, costs and student information necessary for application. During application windows, a link to the OMS CS application site will be posted on the Program Information page</p>

APPENDIX C - NATURAL LANGUAGE PROCESSING (NLP) TOOLS

Google Cloud Natural Language API <https://cloud.google.com/natural-language/>

“The Google Cloud Natural Language API provides natural language understanding technologies to developers, including sentiment analysis, entity analysis, and syntax analysis. This API is part of the larger Cloud Machine Learning API family.” [42] The Google Cloud Natural Language API can be accessed from many programming languages including Python. Google provides a Python library making it easy to use the service from Python programmatically. [43]

Try the API

Do applicants need to take the GRE?

ANALYZE

[See supported languages](#)

Entities	Sentiment	Syntax																																
Do (applicants) ₁ need to take the (GRE) ₂ ?																																		
<p>1. applicants</p> <p>PERSON</p> <p>Sentiment: Score 0 Magnitude 0</p> <p>Salience: 0.86</p>	<p>2. GRE</p> <p>OTHER</p> <p>Sentiment: Score 0 Magnitude 0</p> <p>Salience: 0.14</p>																																	
Entities	Sentiment	Syntax																																
<p><input checked="" type="checkbox"/> Dependency <input checked="" type="checkbox"/> Parse Label <input checked="" type="checkbox"/> Part of Speech <input checked="" type="checkbox"/> Lemma <input checked="" type="checkbox"/> Morphology</p> <table border="1"> <thead> <tr> <th>aux</th> <th>nsubj</th> <th>root</th> <th>aux</th> <th>xcomp</th> <th>det</th> <th>dobj</th> <th>p</th> </tr> </thead> <tbody> <tr> <td>Do</td> <td>applicants</td> <td>need</td> <td>to</td> <td>take</td> <td>the</td> <td>GRE</td> <td>?</td> </tr> <tr> <td>VERB</td> <td>NOUN</td> <td>VERB</td> <td>PRT</td> <td>VERB</td> <td>DET</td> <td>NOUN</td> <td>PUNCT</td> </tr> <tr> <td>mood=INDICATIVE tense=PRESENT</td> <td>number=PLURAL</td> <td></td> <td></td> <td></td> <td></td> <td>number=SINGULAR proper=PROPER</td> <td></td> </tr> </tbody> </table>			aux	nsubj	root	aux	xcomp	det	dobj	p	Do	applicants	need	to	take	the	GRE	?	VERB	NOUN	VERB	PRT	VERB	DET	NOUN	PUNCT	mood=INDICATIVE tense=PRESENT	number=PLURAL					number=SINGULAR proper=PROPER	
aux	nsubj	root	aux	xcomp	det	dobj	p																											
Do	applicants	need	to	take	the	GRE	?																											
VERB	NOUN	VERB	PRT	VERB	DET	NOUN	PUNCT																											
mood=INDICATIVE tense=PRESENT	number=PLURAL					number=SINGULAR proper=PROPER																												

Figure C1. Natural Language Processing example using Google Natural Language console

Figure C1 shows the output of the Google Natural Language system which has similar capabilities to the NLTK and TextBlob Python libraries or the CoreNLP and OpenNLP Java libraries. Google Natural Language performs the following analysis on input text: entity analysis, syntactic analysis and sentiment analysis. [42]

Entity analysis extracts entities (proper and common nouns) from the input text and returns the salience (importance) of each word. Syntactic analysis extracts linguistic information, breaking up the given text into tokens (generally, word boundaries). Sentiment analysis inspects the input text and identifies the emotional opinion within the text: positive, negative, or neutral. This analysis is performed on individual nouns and verbs in the sentence and on the whole sentence. The emotional opinion of the text can be used as a proxy of the text writers attitude when the text was written. [42]

Entity analysis

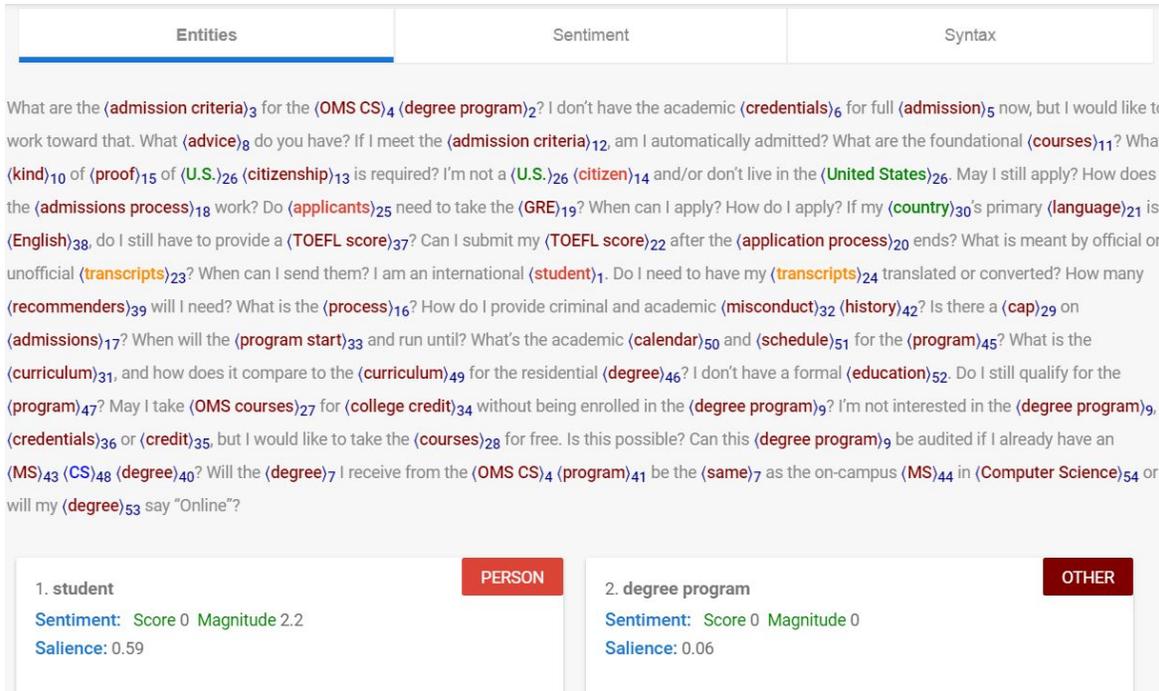


Figure C2. Entities extracted from all questions in appendix A.

The entities in the text, “Do applicants need to take the GRE?” are ‘applicants’ and ‘GRE’ (shown in figure B1). Figure C2 shows the entities in all the questions from appendix B. The chatbot can use entities to search for answers in the knowledge base.

Sentiment analysis

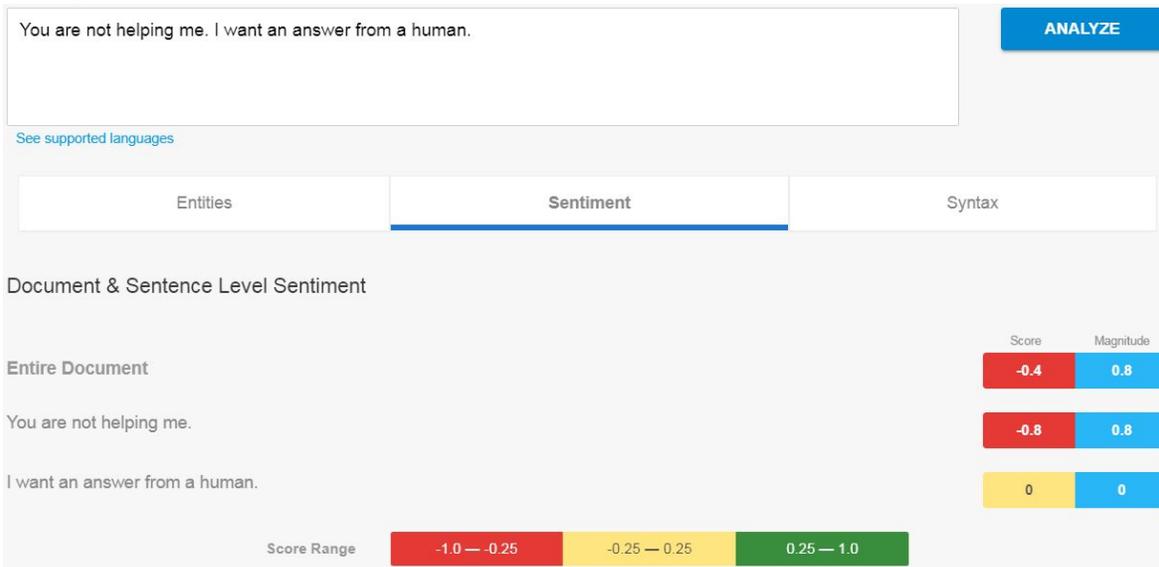


Figure C3. Sentiment analysis

Sentiment analysis provides a tool for estimating the attitude of the text and therefore the attitude of the author. A customer support chatbot can use sentiment analysis to determine when the customer is getting frustrated and a human needs to get involved. As shown in figure C3, the text “you are not helping me. I want an answer from a human.” has a negative sentiment (score is from -1.0 to +1.0). This sentiment score triggers an end to the chatbot conversation and the user is informed a log of the conversation is being sent to a human.

WIT.AI <https://wit.ai>

“Wit.ai can help you parse a message into structured data (Understand) or predict the next action your bot should perform (Converse).” [50] “Wit allows you to extract relevant pieces of information - or entities - from what your users might say to your app.” [50] “An entity is a piece of information you would want to detect from a user input.” [50] Wit is a front end tool based on entities. Trained by providing example user inputs. Although Wit.ai has something called an intent, it’s actually just a custom entity type. [51]

WIT.AI Training Using Understanding

Inbox Understanding Stories ^{beta}

Hello, egregori3!

Welcome to your new Wit app!

Wit allows you to extract relevant pieces of information — or **entities** — from what your users might say to your app.

To get started and create a new entity, you can start teaching your app with the input below, or refer to our [documentation](#).

Test how your app understands a sentence

You can train your app by adding more examples

How many total hours are required?

intent Create new value "hours"

+ Add a new entity

✓ Validate

Your app uses 1 entity

Name	Search Strategy	Values
intent → User-defined entity	trait free-text keywords	courses, foundation course, admission process, admission criteria, GRE, apply

Figure C4. Using ‘understanding’ to train Wit.ai

Wit.ai can be trained using ‘understanding’ or ‘stories’. Figure C4 shows Wit.ai being trained using ‘understanding’. A single user-defined entity called ‘intent’ has been trained with multiple phrases. Each phrase has been assigned with a label (‘value’).

User says	Value
‘How many total hours are required?’	‘hours’
‘Do applicants need to take the GRE?’	‘GRE’
‘How does the admissions process work?’	‘admission process’
‘What are the foundational courses?’	‘foundation course’

'When can I apply?'	'apply'
'When can people sign up for the courses?'	'courses'

Table C1. Phrases and associated values trained using 'understanding'

intent 

User-defined entity

Search strategy  **trait** free-text keywords

Trait Values

Trait value 
hours
courses
foundation course
admission process
admission criteria
GRE
apply

 Add a new trait

Expressions Filter by: all values

“ Search through your expressions.

Text

“ How many total hours are required?	x
“ When can people sign up for the courses?	x
“ When can I apply?	x
“ Do applicants need to take the GRE?	x
“ How does the admissions process work?	x
“ What are the foundational courses?	x
“ What are the admission criteria for the OMS CS degree program?	x

Figure C5. Results of training with data in Table B1

WIT.AI Test Results 'understanding'

Test Question	Returned Value	Confidence %
How many hours are required to graduate?	hours	84
How many classes are required to graduate?	hours	56
When can I apply?	apply	99
Where can I apply?	apply	99
What is the application process?		
What is the admission process?		
How does the admission process work?		
Is the GRE required?	GRE	99
What is the minimum GRE score required?	GRE	99
What are foundation courses?	foundation course	93
How many foundation courses are required?	hours	55
What are the admission criteria?	admission criteria	67
where do I find the admission criteria?		

Table C2. WIT.AI Test Results 'Understanding'

WIT.AI Training Using Stories

Wit.ai has a new training tool called 'stories'. The tool is currently in beta and poorly documented, but appears to be a powerful tool for creating a conversational chatbot. The 'stories' tool creates a flow based chatbot. The flow is configured using a web tool. During configuration, the user input (question) and corresponding agent response is entered along with branches and jumps in the flow.

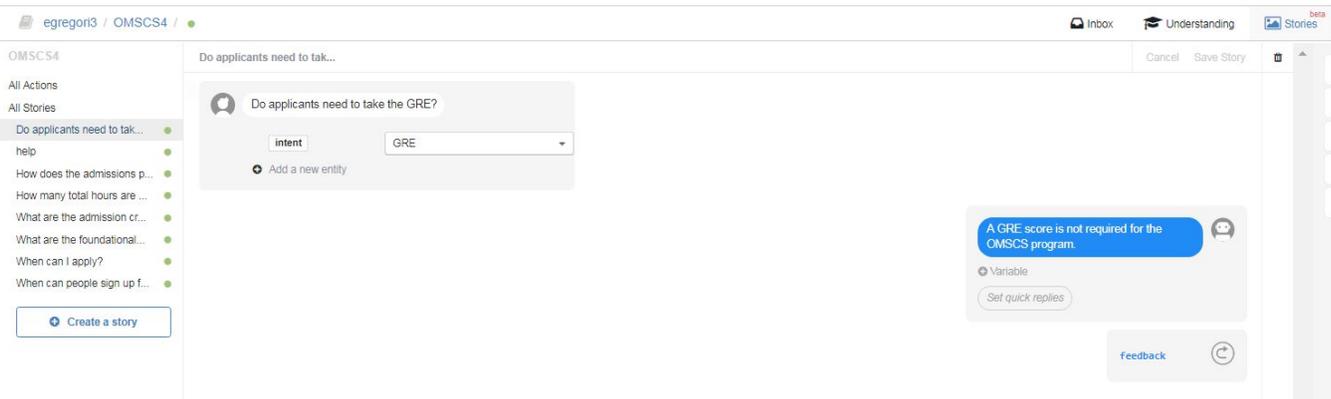


Figure C6. Using 'stories' to train Wit.ai

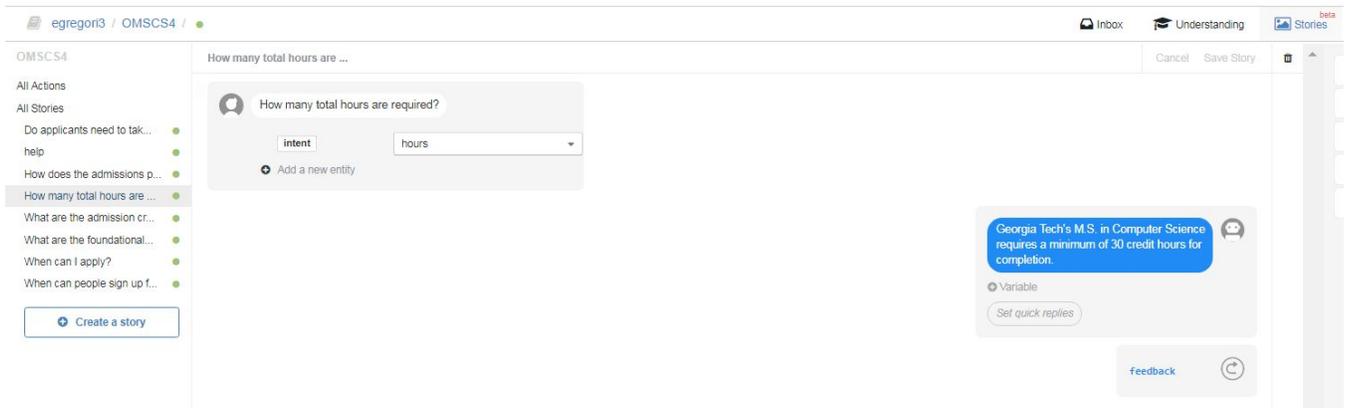


Figure C7. Using 'stories' to train Wit.ai

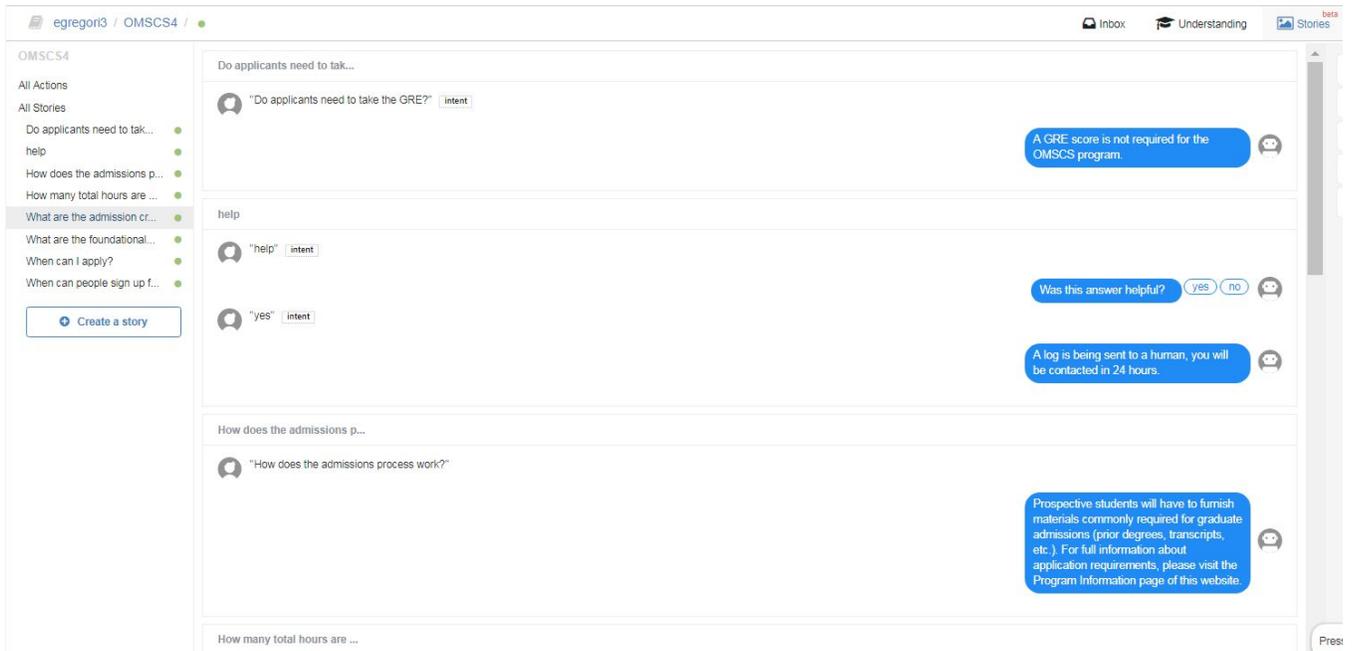


Figure C8. Using 'stories' to train Wit.ai

WIT.AI Test Results 'stories'

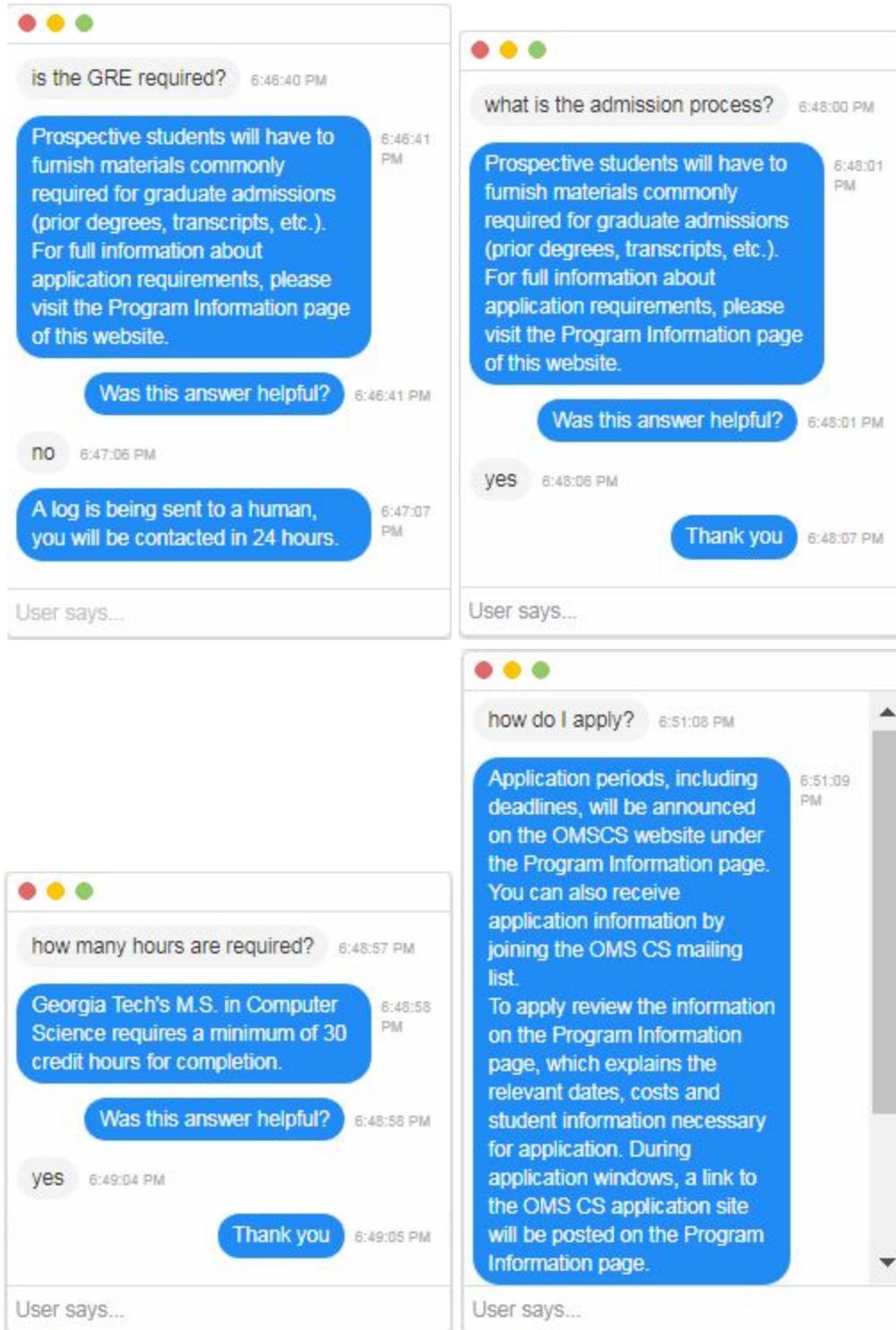


Figure C7. WIT.AI Test Results 'stories'

Microsoft Language Understanding Intelligent Service LUIS
<https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/Home>

“A LUIS app is a place for a developer to define a custom language model. The output of a LUIS app is a web service with an HTTP endpoint that you reference from your client application to add natural language understanding to it. A LUIS app takes a user utterance and extracts intents and entities that correspond to activities in the client application’s logic. Your client application can then take appropriate action based on the user intentions that LUIS recognizes.” [45]

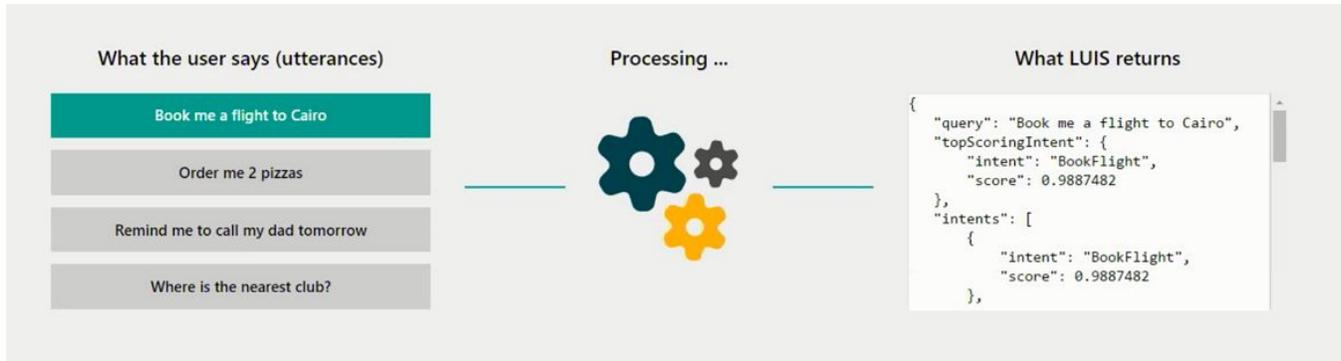


Figure C8. Machine learning is used to classify user utterances (text) into intents [45]

LUIS Intents

LUIS uses machine learning to extract the intent from the user’s text. “Intents are like verbs in a sentence. An intent represents actions the user wants to perform. It is a purpose or goal expressed in a user’s input, ...” [45] “An intent represents a task or action the user wants to perform. It is a purpose or goal expressed in a user’s input, or utterances. Intents match user requests with the actions that should be taken by your app, so you must add intents to help your app understand user requests and respond to them.” [45] Intents are trained by entering multiple utterances with similar meaning into a classifier. After training, the classifier is used to map similar text to actions.

LUIS Entities

“If intents are verbs, then entities are nouns. An entity represents an instance of a class of object that is relevant to a user’s intent.” [45] By recognizing the entities that are mentioned in the user’s input, LUIS helps you choose the specific actions to take to fulfill an intent. [45] Entities are used to clarify the intent.

gre

Here you are in full control of this intent; you can manage its utterances, used entities and suggested utterances ... [Learn more](#)

Utterances (4) Entities in use Suggested utterances

Type a new utterance & press Enter ...	
Utterance text	Predicted Intent
<input type="checkbox"/> is the gre required ?	1 gre
<input type="checkbox"/> what is the minimum gre score required for admission ?	1 gre
<input type="checkbox"/> do i need to take the gre ?	1 gre
<input type="checkbox"/> do applicants need to take the gre ?	1 gre

Figure C9. Training a LUIS intent labeled ‘gre’ with utterances

Test Question	Returned Intent Tag	Confidence %
How many hours are required to graduate?	curriculum	21
How many classes are required to graduate?	curriculum	16
When can I apply?	application	100
Where can I apply?	application	100
What is the application process?	curriculum	6
What is the admission process?	curriculum	5
How does the admission process work?	curriculum	7
Is the GRE required?	gre	100
What is the minimum GRE score required?	gre	100
What are foundation courses?	foundation_courses	100
How many foundation courses are required?	foundation_courses	57
What are the admission criteria?	admission	99
where do I find the admission criteria?	admission	87

Table C3. Testing LUIS after training with utterances and intents from appendix B.

api.ai Agents <https://api.ai/>

“Agents are best described as NLU (Natural Language Understanding) modules. These can be included in your app, product, or service and transforms natural user requests into actionable data. This transformation occurs when a user input matches one of the intents inside your agent. Intents are the predefined or developer-defined components of agents that process a user’s request. Agents can also be designed to manage a conversation flow in a specific way. This can be done with the help of contexts, intent priorities, slot filling, responsibilities, and fulfillment via webhook.” [44]

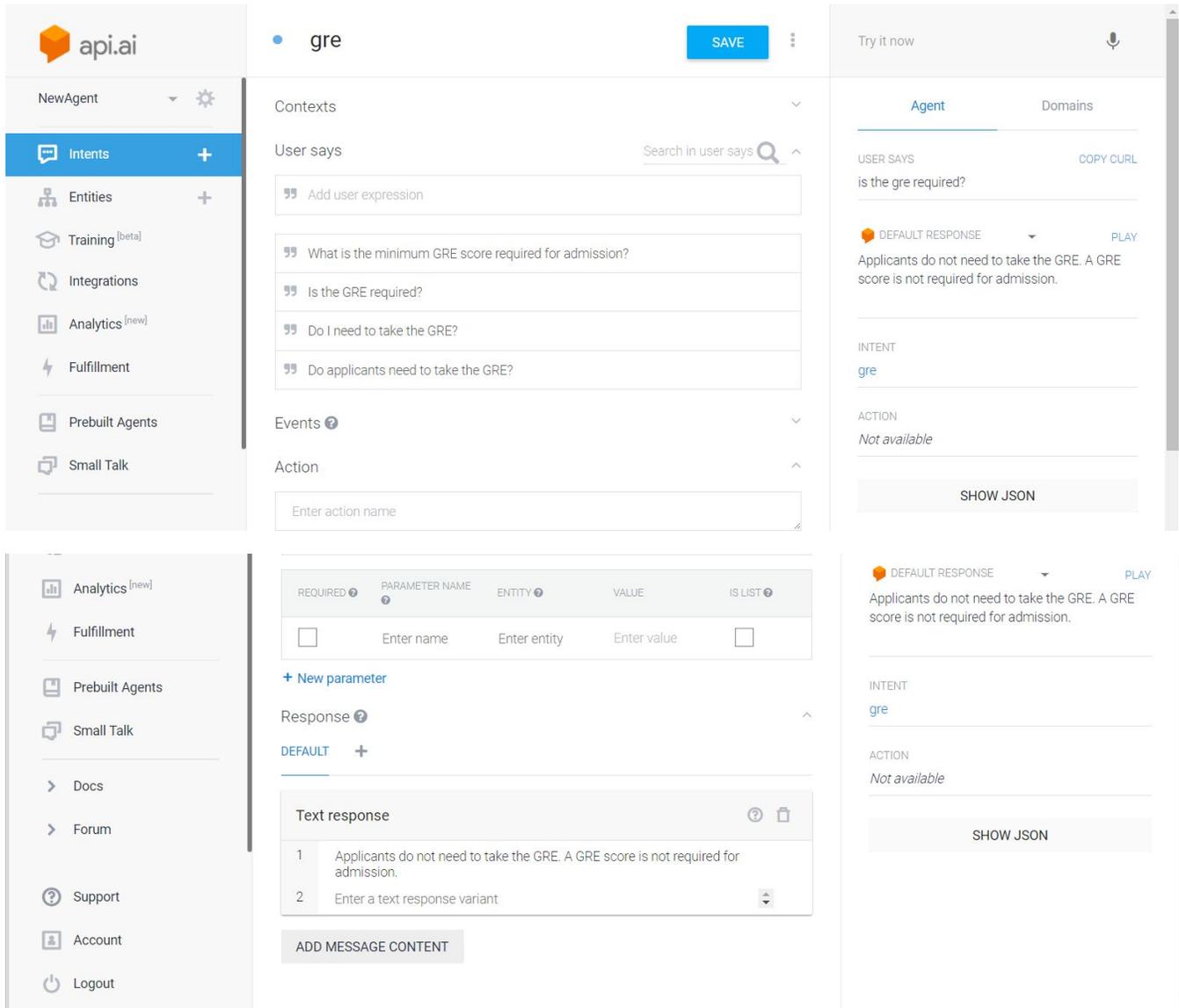


Figure C10. API.ai Agent Console. Training ‘gre’ intent from appendix B.

Figure C10 shows the API.ai Console. An intent name ‘gre’ is being configured with utterances from appendix B. API.ai supports assigning responses to intents. After assigning utterances and responses to intent, the agent is trained.

Test Question	Returned Intent Tag	Confidence %
How many hours are required to graduate?	curriculum	68
How many classes are required to graduate?	curriculum	40
When can I apply?	application	100
Where can I apply?	application	100
What is the application process?	failed	
What is the admission process?	failed	
How does the admission process work?	failed	
Is the GRE required?	gre	100
What is the minimum GRE score required?	gre	84
What are foundation courses?	foundation_courses	74
How many foundation courses are required?	foundation_courses	41
What are the admission criteria?	admission	49
where do I find the admission criteria?	application	35

Table C4. Testing APL.ai after training with utterances and intents from appendix B.

The screenshot shows the API.ai training interface. On the left is a sidebar with navigation options: NewAgent, Intents, Entities, Training (beta), Integrations, Analytics (new), Fulfillment, Prebuilt Agents, Small Talk, Support, Account, and Logout. The main window is titled 'is the gre required?' and includes a 'CLOSE' button in the top right. Below the title, it displays statistics: '14 REQUESTS', '3 UNMATCHED', and '9 MINS'. An 'APPROVE' button is located in the top right corner of the main content area. The main content area contains a list of four user queries, each with a corresponding intent assignment status:

- USER SAYS: What is the application process? INTENT: Click to assign (failed)
- USER SAYS: What is the admission process? INTENT: Click to assign (failed)
- USER SAYS: How does the admission process work? INTENT: Click to assign (failed)
- USER SAYS: Is the GRE required? INTENT: gre (successful)

The response for the successful query is: 'Applicants do not need to take the GRE. A GRE score is not required for admission.' The interface also features a 'SHOW' button at the bottom right.

Figure C11. Re-training API.ai on the responses it failed.

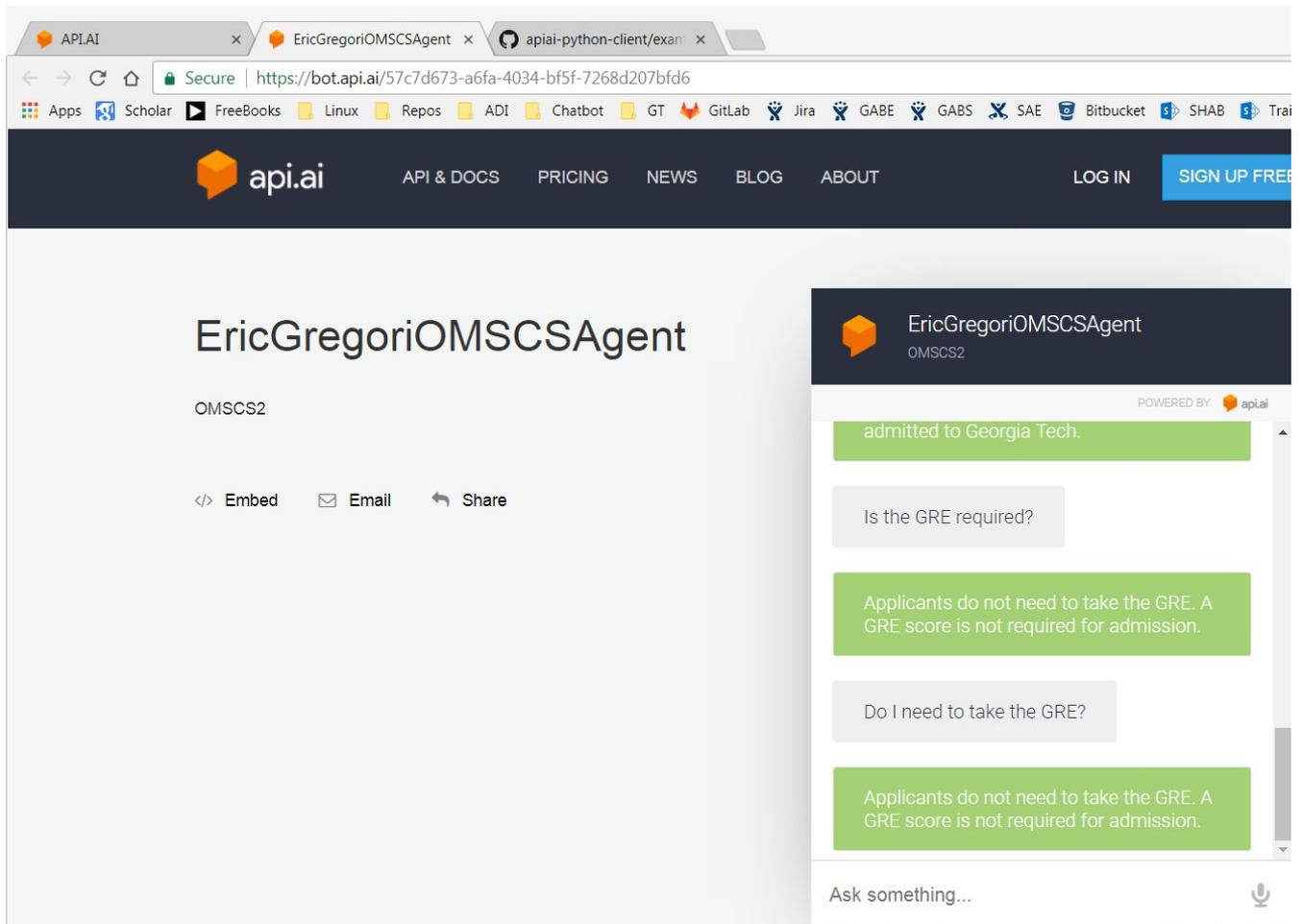


Figure C12. Web Demo One Click Integration.

<https://bot.api.ai/57c7d673-a6fa-4034-bf5f-7268d207bfd6>

AMAZON LEX <https://aws.amazon.com/lex/>

“Amazon Lex is a service for building conversational interfaces into any application using voice and text. Amazon Lex provides the advanced deep learning functionalities of automatic speech recognition (ASR) for converting speech to text, and natural language understanding (NLU) to recognize the intent of the text, to enable you to build applications with highly engaging user experiences and lifelike conversational interactions. With Amazon Lex, the same deep learning technologies that power Amazon Alexa are now available to any developer, enabling you to quickly and easily build sophisticated, natural language, conversational bots (“chatbots”).” [40]

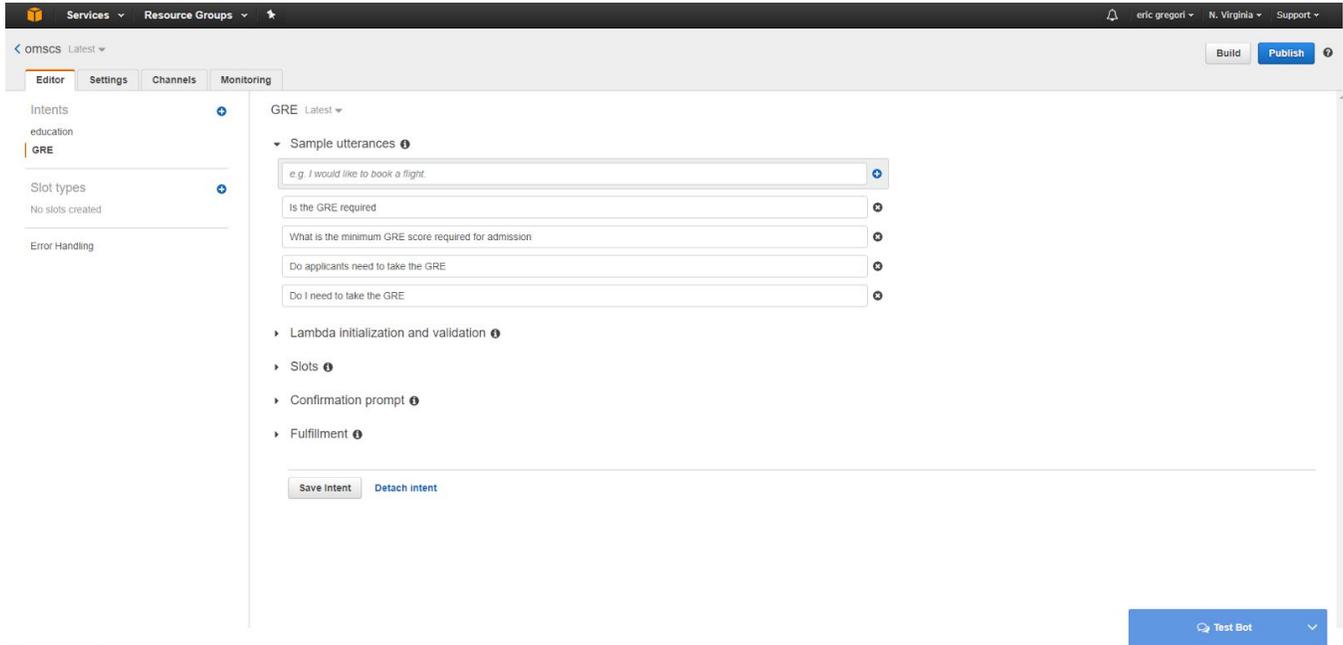


Figure C13. Amazon Lex Console. Entering utterances for ‘gre’ intent.

Test Question	Returned Intent Tag	Confidence %
How many hours are required to graduate?	curriculum	NA
How many classes are required to graduate?	curriculum	NA
When can I apply?	application	NA
Where can I apply?	application	NA
What is the application process?	gre	NA
What is the admission process?	gre	NA
How does the admission process work?	failed	NA
Is the GRE required?	gre	NA
What is the minimum GRE score required?	gre	NA
What are foundation courses?	failed	NA
How many foundation courses are required?	curriculum	NA
What are the admission criteria?	education	NA
Where do I find the admission criteria?	application	NA

Table C5. Testing Lex after training with utterances and intents from appendix B.

APPENDIX D - MICROSOFT AZURE QNA MAKER TOOL - [QNAMAHER.AI](https://qnamaker.ai)

“Microsoft QnA Maker is a free, easy-to-use, REST API and web-based service that trains AI to respond to user's questions in a more natural, conversational way. Compatible across development platforms, hosting services, and channels, QnA Maker is the only question and answer service with a graphical user interface—meaning you don’t need to be a developer to train, manage, and use it for a wide range of solutions.” [9]

KNOWLEDGE BASE 67 QnA pairs		+ Add new QnA pair
4	What are the foundational courses?	They are indicated in the course listing on the [Program Information](https://www.omscs.gatech.edu/program/) page of this website.
5	What kind of proof of U.S. citizenship is required?	For more information on accepted proof of citizenship, please visit [the Georgia Tech graduate admissions website] (http://www.grad.gatech.edu/lawful-presence).
6	I'm not a U.S. citizen and/or don't live in the United States. May I still apply?	Yes, subject to [U.S. export control policy](http://researchintegrity.gatech.edu/about-export/export_overview/).
7	How does the admissions process work?	Prospective students will have to furnish materials commonly required for graduate admissions (prior degrees, transcripts, etc.). For full information about application requirements, please visit the [Program Information] (https://www.omscs.gatech.edu/program/) page of this website.
8	Do applicants need to take the GRE?	No.
9	When can I apply?	Application periods, including deadlines, will be announced on the OMSCS website under the [Program Information](https://www.omscs.gatech.edu/program-information) page. You can also receive application

Figure C1. A portion of the knowledge base automatically created by Microsoft QnA Maker from FAQ in appendix A

The screenshot displays three chat-style interactions with a knowledge base. Each interaction shows a user question in a grey bubble and an AI response in a light blue bubble. The AI responses are: 'No.' for 'do i need to take the GRE?', 'No.' for 'is the GRE required?', and a detailed paragraph for 'what is the minimum GRE score?'.

Figure C2. Example requests and responses from knowledge base

QnA Maker automatically converts a FAQ of questions and answers (appendix A) into a knowledge base (figure C1). The knowledge base is intelligent in that the request does not have to match the FAQ question directly. As shown in figure C2, the request does not need to be worded the same as it is in the FAQ to get the correct response. The FAQ words the question as, “Do applicants need to take the GRE?” The knowledge base returned the correct response when the request wording was, “Do I need to take the GRE?” or “Is the GRE required?” Note when worded as, “What is the minimum GRE score?” the knowledge base returned an incorrect response.

The knowledge base can be fixed manually by either adding new question and answer pairs or by assigning multiple questions of different wording to the same answer. For example, to correct the response to, “What is the minimum GRE score?” the knowledge base can be updated with the correct response, “No GRE score is required.”

Using the Knowledge Base

Success! Your service has been deployed. What's next?

You can always find the deployment details in your service's settings.

Use the below HTTP request to build your bot. [Learn how.](#)

Sample HTTP request	POST /knowledgebases/[REDACTED]/generateAnswer Host: https://westus.api.cognitive.microsoft.com/qnamaker/v2.0 Ocp-Apim-Subscription-Key: [REDACTED] Content-Type: application/json { "question": "hi" }
---------------------	---

Need to fine-tune and refine? Go back and keep editing your service.

[Edit Service](#)

Figure C3. Accessing the knowledge base

```
import requests

class QnAMaker(object):
    host = "https://westus.api.cognitive.microsoft.com/qnamaker/v2.0"
    answers = None
    answerIndex = 0

    def __init__(self, KBid, SubKey):
        self.KnowledgeBaseID = KBid
        self.SubscriptionKey = SubKey

    def Request(self, Question):
        self.answers = None
        url = "/knowledgebases/"
        url += self.KnowledgeBaseID
        url += "/generateAnswer"
        header = {'Ocp-Apim-Subscription-Key': self.SubscriptionKey}
        header.update( {'Content-Type': 'application/json'} )
        data = {'question': Question}
        r = requests.post(self.host+url, headers=header, json=data)
        # r.text = {"answers":[{"answer":"Hello","questions":["Hi"],"score":100.0}] }
        answers = r.json()
        if( "answers" in answers ):
            self.answers = answers["answers"] # Extract list of QnA dictionaries
            self.answerIndex = 0
            return r.status_code
        else:
            print "!ERROR!"
            print r.text
            return 0

# Returns list of QnA dictionaries
def GetAnswerList(self):
    return self.answers

# Returns next QnA dictionary in list
def GetNextAnswer(self):
    if( self.answerIndex < len(self.answers) ):
        qnaDict = self.answers[self.answerIndex]
        self.answerIndex += 1
        return qnaDict
    return None
```

Code Block D1. Python class for querying QnA Knowledge base

```

import QnAMaker

TestQnAMaker = QnAMaker.QnAMaker( KBid='Knowledge Base ID', \
                                   SubKey='insert Subscription Key')

ReturnCode = TestQnAMaker.Request('insert question here')
print " - ReturnCode: ", ReturnCode
if( ReturnCode == 200 ):
    answer = TestQnAMaker.GetNextAnswer()
    print "Score: ", answer['score']
    print "Answer:"
    print answer['answer']

```

Code Block D2. Portion of Python code to test Knowledge Base

Once published the knowledge based can be accessed using standard HTTP methods.. Figure D3 is an example of how to request an answer to ‘hi’ from the knowledge base using HTTP. A knowledge base ID and subscription key is used to access the knowledge base. The ID and key can be found after the knowledge base is published in the sample HTTP request as shown in figure D3 (blacked out).

The Python class shown in code block D1 is used to access the knowledge base. Code block D2 is an example of how to use the class. There is a limit to the rate at which requests can be made to the knowledge base. For this test there was a 30 second delay between questions.

Test Results

Question	Microsoft QnA Maker Response	Score
Is a GRE required?	No	44.6
Do I need to take the GRE?	No	50.2
Do I need a bachelor's degree?	Preferred qualifications for admitted OMS CS students are an undergraduate degree in computer science or related field (typically mathematics, computer engineering or electrical engineering) from an accredited institution with a cumulative GPA of 3.0 or higher. Applicants who do not meet these criteria will be evaluated on a case-by-case basis; significant professional or other work experience with supporting recommendations may qualify as an adequate substitute for the appropriate academic credentials, however work experience will not take the place of an undergraduate degree. Georgia Tech will not admit applicants into the OMS CS degree program without the minimum qualifications for success. The Georgia Tech minimum criteria used in determining each applicant’s eligibility for consideration shall include: Evidence of award of a bachelor’s degree or its equivalent (prior to matriculation) from a recognized institution, demonstrated academic excellence, and evidence of preparation in their chosen field sufficient to ensure successful graduate study; and For international applicants, satisfactory scores on the Test of English as a Foreign Language (TOEFL).	34.5
Is the course material available for free?	Course offerings are listed on the [Program Information](https://www.omscs.gatech.edu/program/) page.	33.2
Can the degree be audited?	Not at first. We will provide additional program tracks following successful launch of the degree program, including possibly an audit track.	98.0
Does the degree mention online?	Your diploma will read ‘Master of Science in Computer Science,’ exactly the same as those of on-campus graduates. There will be no ‘online’ designation for the degrees of OMS CS graduates.	34.9
How many foundation courses do I need?	Students may take only up to two courses until they pass their foundational requirement. Once fully admitted, the maximum number of hours a student may take is nine hours a semester (three courses). Be advised that each course is just as rigorous as its on-campus equivalent. We recommend that OMS CS degree-seeking students working full-time while taking courses enroll in no more than two courses simultaneously.	42.8
What are the current	Specialization offerings are listed on the	70.4

specializations?	[Specializations](https://www.omscs.gatech.edu/program-info/specializations) page of this website.	
What classes are offered this semester?	Students may take only up to two courses until they pass their foundational requirement. Once fully admitted, the maximum number of hours a student may take is nine hours a semester (three courses). Be advised that each course is just as rigorous as its on-campus equivalent. We recommend that OMS CS degree-seeking students working full-time while taking courses enroll in no more than two courses simultaneously.	31.8
Are classes offered in the summer?	No. International students applying to OMS will not be offered visas, hence they will not qualify for OPT.	26.2
How do I apply?	First, review the information on the [Program Information](https://www.omscs.gatech.edu/program/) page, which explains the relevant dates, costs and student information necessary for application. During application windows, a link to the OMS CS application site will be posted on the Program Information page.	99.0
When do I apply?	First, review the information on the [Program Information](https://www.omscs.gatech.edu/program/) page, which explains the relevant dates, costs and student information necessary for application. During application windows, a link to the OMS CS application site will be posted on the Program Information page.	98.0
What do I need to apply?	First, review the information on the [Program Information](https://www.omscs.gatech.edu/program/) page, which explains the relevant dates, costs and student information necessary for application. During application windows, a link to the OMS CS application site will be posted on the Program Information page.	48.6
How does admissions work?	Prospective students will have to furnish materials commonly required for graduate admissions (prior degrees, transcripts, etc.). For full information about application requirements, please visit the [Program Information](https://www.omscs.gatech.edu/program/) page of this website.	72.8
How many recommendations do I need?	Applicants must procure recommendations from three individuals. As part of the application, applicants will provide their recommenders' email addresses. Those individuals will be contacted through email and provided instructions on how to submit their recommendations online. A physical letter will not be required.	51.8
How much do classes cost?	Exact cost will depend on how quickly students complete the program. We anticipate that working students will take an average of two courses per term, resulting in a total program cost of about \$6,600 over five terms. Students who complete their programs more quickly will pay less; those who take longer will pay more.	70.9
How do I transfer?	Graduate students may not "transfer" to Georgia Tech in the same manner that undergraduates can transfer; however, you are free to apply to the OMS program. The Institute limits the number of M.S. transfer credits to six hours, subject to the approval of the College of Computing.	86.5

Table C1. Microsoft QnA Maker test results when trained with FAQ from appendix A

Incorrectly Answered Question	Score	FAQ - Question	Correct Answer
Is the course material available for free?	33.2	I'm not interested in the degree program, credentials or credit, but I would like to take the courses for free. Is this possible?	Yes, the OMS courses are available for free on the Udacity website. These free courses do not carry college credit and offer limited academic support.
What classes are offered this semester?	31.8	What courses are available?	Course offerings are listed on the Program Information page.
Are classes offered in the summer?	26.2	Not available in FAQ	Not available in FAQ

Table D2. Incorrectly answered questions and the FAQ entry with the correct answer

Table D1 shows the responses and scores to the test questions. Of the 17 questions, only 3 were answered incorrectly. Table D2 shows the 3 questions Microsoft QnA Maker answered incorrectly along with the FAQ entries with the correct answers. The scores for the incorrect answers are all below 34.0. "What classes are available?" resulted in a correct response from QnA Maker without additional training. Unfortunately, the responses and scores from QnA Maker change every time the test is run. This behavior is not understood or documented. More research needs to be done to understand this anomaly.

Conversational Dialog

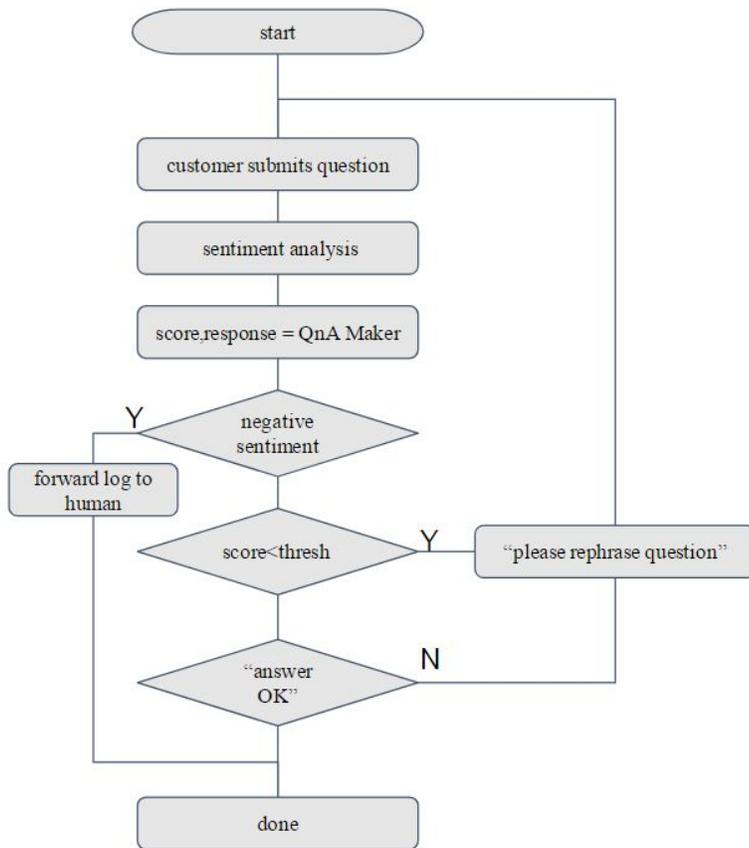


Figure D4. Simple conversation chatbot using QnA Maker with sentiment analysis

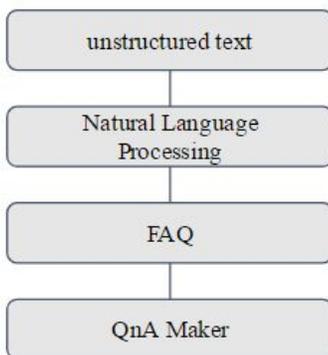
QnA Maker does not support conversational dialogs directly. The front end AI manages the conversation using the scores returned from QnA Maker. If QnA Maker returns a score less than a threshold, the front end AI agent asks the user to either rephrase the question or provide additional information.

Figure D4 is a flowchart for a very simple conversational chatbot using QnA Maker. If the question sentiment is negative indicating the customer is getting upset, the conversation log is sent to a human. If the QnA Maker score is less than a threshold, the customer is asked to rephrase the question. Finally, if the score is greater than the threshold, the customer is asked if the answer is OK. If the answer is OK, the dialog is done. If not, the customer is asked to rephrase the question.

The flowchart does not show a question counter but one is required. As described above, customers lose interest after 4 Q/A pairs. The counter would stop the conversation after 4 attempts and forward the conversation log to a human.

When the conversation log is forwarded to a human the user is informed to expect an answer within 24 hours.

Unstructured Domain Data



QnA Maker currently can only consume data in the form of Question/Answer pairs (FAQ's). As shown in figure D5, unstructured domain data in the form of unstructured text must be converted into Question/Answer pairs by a separate tool. Natural Language Processing (NLP) can be used to convert unstructured text into a FAQ. The FAQ can then be input into QnA Maker.

Initial research into unstructured text to FAQ converts yields no results. This could be an interesting area of research. A tool that can convert unstructured text into FAQ's could have value in more areas than just chatbots.

Figure D5. Unstructured text to FAQ converter

APPENDIX E - TOOLS TESTING

User Interface

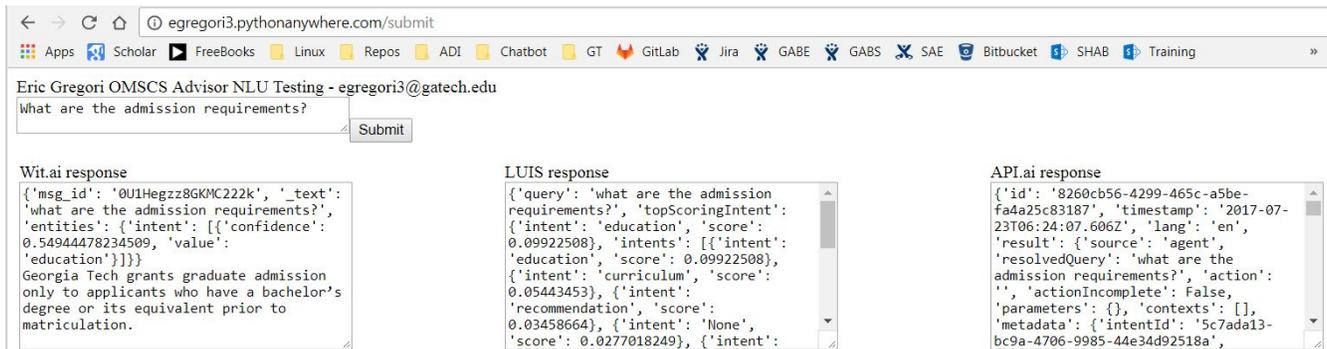


Figure E1. User interface for testing Wit.ai, LUIS, Api.ai

The test tool uses a web user interface shown in figure in figure E1. The user enters a question in the text box and clicks submit. The question is sent to the Wit.ai, LUIS, and Api.ai NLU tools. The response from each tool is displayed in the corresponding labeled text box. In addition to the raw response from the tool, the response assigned to the returned intent is also displayed.

Pythonanywhere does not whitelist the Amazon Lex server so Amazon Lex was not included in this test tool. Amazon Lex was tested using the Amazon Lex dashboard

Block Diagram

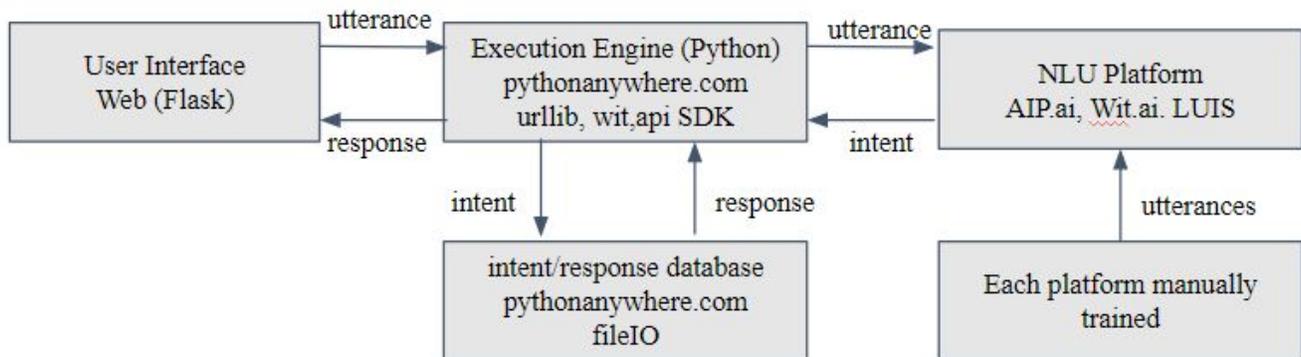


Figure E2. NLU Testing Tool

Figure E1 shows the user interface for the NLU test tool developed by the author for evaluating the top three NLU platforms: Wit.ai, API.ai, LUIS. The tool is based on Python Flask on the PythonAnywhere platform. Amazon Lex was also evaluated, but not as part of the tool. Amazon Lex was evaluated using the Lex tester provided in the dashboard.

Figure E2 is a block diagram of the test tool. The execution engine builds the HTML for the user interface and serves the web page to the web browser using the Python Flask HTTP server library. The user enters a question (utterance) into the textbox in the web browser and clicks submit. The utterance is submitted to the execution engine by the web browser via HTTP. This triggers the execution of a Python function in the execution engine.

The Python function submits the utterance to Wit.ai, API.ai, and LUIS. The raw JSON response from each platform is displayed in the appropriate test box. The tester tool files can be found on Github here:

https://github.com/egregori3/OMSCSAdvisor_Flask_NLUTester

Microsoft LUIS Configuration. <https://www.luis.ai/applications>

App Id: 5dd6e368-995f-4c80-ad5c-73307670cf7d

ⓘ You have no entities yet. Entities represent key data relevant to the app and may be essential for your app to perform its task. Think of the main entities of your app and start creating them.

[Create an entity](#) [Next tip](#)

App status
 Last train: Jul 16, 2017, 5:58:48 PM Last published: Jul 23, 2017, 2:23:02 AM

Intent Count	Entity Count	List Entity Count	Labeled Utterances Count
10 / 80	0 / 30	0 / 50	17

Endpoint Hits Per Period
PER DAY (LAST WEEK)

Total Endpoint Hits
SINCE APP CREATION

1

Intent Breakdown
ON LABELED UTTERANCES

Entity Breakdown
ON LABELED UTTERANCES

No entities in app

Intents

A listing of intents in the application. Click an intent to view/edit its details, or add a new intent ... [Learn more](#)

Add Intent
Add prebuilt domain intents

🔍

Intent Name ↓	Utterances	ⓘ 🗑️
None	0	
admission	1	ⓘ 🗑️
application	2	ⓘ 🗑️
curriculum	2	ⓘ 🗑️
education	2	ⓘ 🗑️
foundation_courses	1	ⓘ 🗑️
gre	4	ⓘ 🗑️
recommendation	2	ⓘ 🗑️
toefl	1	ⓘ 🗑️
transcript	2	ⓘ 🗑️

intent



User-defined entity

Search strategy ⓘ

trait

free-text

keywords

Trait Values

Trait value ⓘ
application
foundation
curriculum
toefl
transcript
admission
recommendation
education
gre

⊕ Add a new trait

Expressions

Filter by:

all values

“ Search through your expressions.
Text
“ when can I apply? ×
“ Where do I apply? ×
“ What are the foundational courses? ×
“ What are the admission criteria for the OMS CS degree program? ×
“ How many total hours are required? ×
“ What is the curriculum, and how does it compare to the curriculum for the residential degree? ×
“ If my country's primary language is English, do I still have to provide a TOEFL score? ×
“ What is meant by official or unofficial transcripts? ×
“ Do I need to have my transcripts translated or converted? ×
“ Do I need recommendations? ×
“ How many recommenders will I need? ×
“ What level of formal education is required? ×
“ What are the minimum education requirements? ×
“ What is the minimum GRE score required for admission? ×
“ Is the GRE required? ×
“ Do I need to take the GRE? ×
“ Do applicants need to take the GRE? ×

EricGregoriOMSCSAgent

SAVE



- General
- ML Settings
- Export and Import
- Share



DESCRIPTION

OMSCS2

LANGUAGE ?

English

DEFAULT TIME ZONE

(GMT-4:00) America/...

GOOGLE PROJECT

Project ID	newagent-1e254
------------	----------------

Intents ?

CREATE INTENT



Search intents 🔍

- admission
- application
- curriculum
- 🔖 Default Fallback Intent
- Default Welcome Intent
- education
- foundation_courses
- gre
- recommendation
- toefl
- transcript

APPENDIX F - OMSCS CHATBOT - [HTTP://EGREGORI3.PYTHONANYWHERE.COM](http://EGREGORI3.PYTHONANYWHERE.COM)

Source Code - https://github.com/egregori3/ChatBotTool_Phase1

OMSCS Chatbot Screenshots



Figure F1. On the main page, either the NLU test tool or the OMSCS Advisor can be selected

**Eric Gregori OMSCS Advisor Conversational Agent - egregori3@gatech.edu
Ask me about OMSCS admissions or curriculum**

Figure F2. OMSCS Advisor start of dialog

**Eric Gregori OMSCS Advisor Conversational Agent - egregori3@gatech.edu
Ask about OMSCS admissions or curriculum.**

```
sessionID:b5c1c756-0303-4abf-8e34-10f1e6b90213
userInput:is the gre required?
intent:gre
confidence:1.0
contexts:[{'name': 'answered', 'parameters': {}, 'lifespan': 1}]

Is this the answer you are looking for (yes/no)? - Applicants do not need to
take the GRE. A GRE score is not required for admission.

sessionID:b5c1c756-0303-4abf-8e34-10f1e6b90213
userInput:yes
intent:positive
confidence:1.0
contexts:[{'name': 'done', 'parameters': {}, 'lifespan': 1}]

Thanks for using the OMSCS advisor. If you have any additional question,
please ask.
```

Figure F3.

User asked, “Is the GRE required?” - context = answered

User answers ‘yes’ to, “Is this the answer you are looking for?” context = done

Success

Ask about OMSCS admissions or curriculum.

Enter text and click Submit

```
Session Time remaining:82.28754091262817 seconds
sessionID:455358d5-3cb5-4ea9-a505-1e4713563bb3
userInput:Who is Eric Gregori?
intent:Default Fallback Intent
confidence:1.0
contexts:[{'name': 'notanswered', 'parameters': {}, 'lifespan': 5}]

I have not been trained on that information. I can only answer questions about
OMSCS admissions and curriculum at this time.
```

Ask about OMSCS admissions or curriculum.

Enter text and click Submit

```
Session Time remaining:91.48532032966614 seconds
sessionID:3c65e00e-ff19-4792-a0fd-f0525e774105
userInput:no
intent:negative
confidence:1.0
contexts:[{'name': 'failed', 'parameters': {}, 'lifespan': 4}]

Please reword the question. I can help with OMSCS admissions and curriculum.

sessionID:3c65e00e-ff19-4792-a0fd-f0525e774105
userInput:where can I take the GRE?
intent:gre
confidence:0.5
contexts:[{'name': 'answered', 'parameters': {}, 'lifespan': 1}]

Is this the answer you are looking for (yes/no)? - Applicants do not need to
take the GRE. A GRE score is not required for admission.
```

Figure F4.

**User asked, "Who is Eric Gregori?"
context = notanswered**

Figure F5.

**User asked, "Where can I take the GRE?"
context = answered
User answers 'no' to, "Is this the answer you are looking for?"
context = failed**

API.ai Json Response Definition

Name	Type	Description
id	String	Unique identifier of the result.
timestamp	String	Date and time of the request in UTC timezone using ISO-8601 format.
lang	String	Agent's language.
result		Contains the results of the natural language processing.
source	String	Source of the answer. Could be "agent" if the response was retrieved from the agent.
resolvedQuery	String	The query that was used to produce this result.
action	String	An action to take. Example: turn on
actionIncomplete	Boolean	true if the triggered intent has required parameters and not all the required parameter values have been collected false if all required parameter values have been collected or if the triggered intent doesn't contain any required parameters
parameters	Object	Object consisting of "parameter_name":"parameter_value" pairs. Example: {"device": "computer" }
contexts	Array	Array of context objects with the fields "name", "parameters", "lifespan".
name	String	Context name.
parameters	Object	Object consisting of "parameter_name":"parameter_value" and "parameter_name.original":"original_parameter_value" pairs.
lifespan	Number	Number of requests after which the context will expire.
fulfillment		Data about text response(s), rich messages, response received from webhook.
speech	String	Text to be pronounced to the user / shown on the screen
messages	Array	Array of message objects
score	Number between 0 and 1	Matching score for the intent.
metadata		Contains data on intents and contexts.
intentId	String	ID of the intent that produced this result.
webhookUsed	String	Indicates whether webhook functionality is enabled in the triggered intent.
webhookForSlotFillingUsed	String	Indicates whether in the triggered intent webhook functionality is enabled for required parameters.
webhookResponseTime	Number	Webhook response time in milliseconds.
intentName	String	Name of the intent that produced this result.
status	Status Object	Contains data on how the request succeeded or failed.
sessionId	String	Session ID.

<https://api.ai/docs/reference/agent/query>