SMU Data Science Review

Volume 1 | Number 1 Article 10

2018

Evaluation of Artificial Intelligence Frameworks

Crystal Todd
Southern Methodist University, ctodd@mail.smu.edu

Ruby Vazquez Pena
Southern Methodist University, rvazquezpena@mail.smu.edu

Raghuram Srinivas
Southern Methodist University, rsrinivas@mail.smu.edu

Evaluation of Artificial Intelligence Frameworks

Crystal Todd[1], Ruby Vazquez Pena[1], and Raghuram Srinivas[1]

[1] Master of Science in Data Science,
Southern Methodist University, Dallas, TX
{ctodd, rvazquezpena, rsrinivas}@smu.edu

Abstract. In this paper, we present a comparative evaluation of three artificial intelligence frameworks, IBM Watson, Amazon Lex, and Microsoft Azure. The comparisons in this paper are in functionality, reliability, usability, efficiency, maintainability, and accessibility of each artificial intelligence frameworks. By comparing and evaluating each framework we have set standardized metrics that help others assess the frameworks for their respective purpose. Data has been gathered to create a virtual admissions assistant for the Southern Methodist University (SMU) Masters in Data Science program. The same data has been used to train three chatbots using IBM Watson, Amazon Lex, and Microsoft Azure. Using the six comparisons, we have set metrics that evaluate each artificial intelligence framework.

1 Introduction

Artificial Intelligence (AI) is not only the theory, but also the development of computer systems ability to perform tasks that normally require human intelligence. AI tries to give computers attributes like visual perception, speech recognition, decision-making, and translation between languages, in essence imitating human attributes. IBM Watson is a top competitor in the AI space as large companies such as H&R Block, Macys, and The Weather Company. It gained traction after beating Ken Jennings and Brad Rutter champions on the TV game show Jeopardy with speed of physically buzzing in and accuracy of answering questions. At that moment in time, this event shined a brighter light on the ethical concerns that involve artificial intelligence. Although those ethical concerns still exist, as time has progressed society has started to utilize artificial intelligence more and more.

Companies like IBM, Amazon, and Microsoft offer products that allow companies to create their own virtual assistants. Chat Bots are virtual assistants trained on large amounts of data, developed to assess questions, and answer these questions in a particular domain accurately. IBM Watson was trained in vast trivia data in order beat the top two human competitors. IBM Watson was used in this event but there is no particular reason why IBM Watson was chosen. Even with minimal knowledge of artificial intelligence frameworks the common assumption of equal frameworks is made. There is little to no research in how Amazon Lex or another AI framework would have done in the same Jeopardy

event compared to IBM Watson. In fact, there is little to no research comparing these AI frameworks in any domain. This can be particularly important for companies that want to choose the best product to answer their customers question or just providing great customer service.

With the identification of the lack of AI frameworks comparisons in mind, we present metrics that can be applied to any AI framework to evaluate the AI frameworks overall fit for a given purpose. These metrics will help companies evaluate existing or new AI frameworks.

2 Frameworks

There are several major players in the Artificial Intelligence field, specializing in cloud computing and commercial advertising. Some frameworks sell themselves as easy out-of-the-box solutions the majority of business analysts will be able to use while others pride themselves in being the tool that data scientists will adore. Thus, businesses across all industries are looking into adding IBM Watson, Amazon Lex, and Microsoft Azure into their future architecture to increase their productivity or profits as well as decrease their out-of-pocket expenses or defects to name a few benefits of AI.

2.1 IBM Watson

IBM Watson is an analytical and question answering computing system that specializes in analyzing natural human language and provides specific answers to complex questions at rapid speeds. IBM intended to advance natural language processing, information retrieval, knowledge representation, automated reasoning, and machine learning with Watson. Many companies utilize IBM Watson and customize it to serve as a customer service agent in their business. IBM advertises Watson as the product that can eliminate customer service calls and inquiries.

IBM Watson can analyze complex unstructured data easily. It also has the capability to contain huge amounts of information and utilize it to answer questions. Famously, IBM Watson was used in Jeopardy! where it easily beat all human players. The difference between IBM Watson and a document search is that IBM Watson takes a question expressed in natural language, seeks to understand it in much greater detail, and returns a precise answer to the question [6].

When answering a question IBM Watson runs through multiple hypotheses, seeks evidence, and scores/ranks the possible answers in order to answer precisely with confidence.

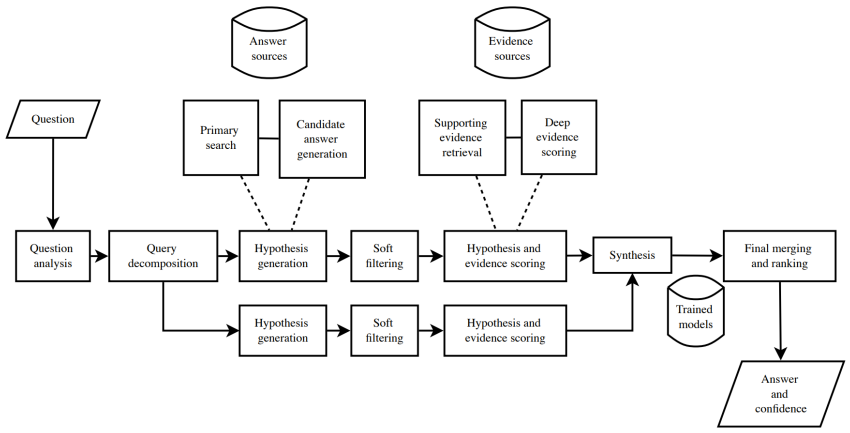

Fig. 1. IBM Watson utilizes IBM's DeepQA software. Above is a high-level architecture of IBM's DeepQA.

Besides answering complex questions, IBM Watson also has the ability to analyze the unstructured data it receives. The ability to analyze this data has been tested by scientists, and IBM Watson has been concluded as the fastest.

2.2 Amazon Lex

Amazon has developed a voice-controlled personal assistant named Alexa in several varieties of speakers. As an insight to its design, Alexa is named after the ancient library of Alexandria where the user speaks a command or a question to the device and it acts accordingly to answer questions, play music, or even turn on the kitchen lights if that functionality is installed. Alexa is made with a natural-language processing system using seven microphones that always listen for the chosen wake word to act upon. Echo is like your dog: Its always listening, but it understands only cookie or walk. Everything else goes right over its head [3].

The backend deep learning algorithms used in Amazon Alexa are available to interested parties through Amazon Web Services. Users can create their own chatbots in a framework called Amazon Lex which allows them to control the development of a chatbot. These newly created chatbots can be migrated into skills for Alexa devices.

2.3 Microsoft Azure

Microsoft (MS) Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a

global network of Microsoft-managed data centers. It runs on a Linux operating system, and provides software as a service (SAAS), platform as a service and infrastructure as a service. Azure supports different programming languages, including Python. With MS Azure, chatbots have the ability to connect with user via various platforms such as: Bing, Skype, Office 365, and Slack. To make the bot as humanly as possible you must use lines of code and add the cognitive services. Cognitive Services gives the bot features like emotion recognition, and face recognition, which are useful when the chatbot is using a webcam with the users. Features like speech recognition, translation, and recommendation on which products your user might want. Chatbots are usually created to serve customers and provide great customer services. The recommendation feature that Cognitive Services includes makes it distinct from other AI frameworks.

On the outside, Azure looks very similar to the other products. However, the focus and strengths of Azure are in enterprise-wide solutions. The security in their product is trusted and verified with more compliance certifications. Microsoft Azure is important to test as many large companies are incorporating this into their businesses and we need to see if it truly is the best solution for these corporations.

3 Attempts of Evaluation

3.1 Siri vs Alexa - Virtual Assistants at the Marriott

At the Marriott, the battle of Siri and Alexa is occurring. The Marriott is trying to roll out either Siri or Alexa in every hotel room. The company is seeking a virtual assistant that will allow their customers to customize everything from the room temperature, lighting, or the entertainment system. Their objective is to see whether the interaction [with the virtual assistant] will be personalized, allow[s] guests familiar with the devices to log into their own accounts, or instead use a standard set of skills relevant to a hotel stay, like getting new reports, checking weather forecasts or calling for an Uber - commands more appropriate for those unfamiliar with the technology [5].

The Marriott provides customer service and is trying to keep that same level of customer services when evaluating both Siri and Alexa. As previously stated, their objective is to allow their clients to do everything they need to do via virtual assistant. Marriott cares about personalization. Marriott will select the device that provides great customer service and ensures accessibility to its customers. Marriott is using personalization and accessibility as metrics in their testing.

3.2 Alexa Metric Dashboard

Amazon has released an Alexa Metrics Dashboard. The purpose of the tool is to drive user retention and the dashboard gives skill developers data points and visualizations on customers, sessions, utterances, and objectives as well as improved data freshness. Since Alexa allows integration with 3rd party applications, developers can utilize these dashboards to view the interaction with

Alexa via their application. The skills dashboard includes five main tabs: an overview that offers a high-level look into a skill's performance, a customer tab that measures customer engagement over selected periods of time, a session tab that highlights what session types are most successful, an utterances tab that measures the quality of customer experience, and an objectives tab to uncover what skill features get the most use.

4 Metrics

We evaluate the Artificial Intelligence device in six ways [2]. They are as follows:

Functionality The measurement of how the device serves its purpose practically.
- Example: Can the artificial intelligence framework actually answer a question in its domain?

Reliability The measurement of how the device performs consistently and is dependent.
- Example: Asking the same question, how many times does it understand that same question? (Varying volume and pitch in the voice)

Usability The measurement of how each device is applicable and useful
- Example: Is the answer the question the best answer? (What is the closest grocery store?)

Efficiency The measurement of how the device performs in competence and productivity while being conservative with computing resources

Maintainability The measurement of how the device overcomes failures and supports maintenance in sustainability of the device

Accessibility The measurement of how the software on the device is available on other operating systems and devices
- Example: Is the software applicable on only Amazon operating systems or does it react appropriately with iOS or Android software as well?

5 Ethics

When talking about AI frameworks, the ethical concerns of human imitation or replacement are always mentioned. There is a perception in society, that as AI frameworks get better at imitating human features, and perfect human intelligence, the need for humans diminishes. The chatbot is in essence a customer services representative that guides customers or users throughout a process. These chatbots not only try to provide accurate responses, but try to provide an experience during the complete process. The aspect of customer services, speech recognition, emotion recognition, and face recognition make human replacement a viable fear. As businesses replace customer service representative with this chatbot, we ignite the fear of human replacement.

Businesses want the best possible chatbot, one that can seamlessly be added to a team and provide the same knowledge and experience that a human can.

Thus metrics to identify what is acceptable from an AI framework is crucial for businesses.

There is also the concern that since a lot of the algorithms used in AI are complicated black box solutions, it is often difficult to determine, much less explain, why it determines one outcome versus another. If a person were to sue a company over an outcome, it would be hard to prove whether the machine was being biased or not in that case. Late July 2017, news stories reported that Facebook recently had to turn off their AI engines when they started speaking to each other in their own made up language because they modified the English language to communicate easier and faster [1]. However, the administrators had no clue how to decipher their communications and shut down the AI. Many writers referenced the idea of Skynet from the Terminator movie franchise that wanted to end human existence for fear that humans would try to destroy it first.

Ethically, the idea that AI could go rogue away from the original intent is a concern as the use cases for AI are limitless. To demonstrate this, one possible use for AI could be to determine the sentencing of a criminal, determining if they deserve prison time, how long, how much for bail, and if they should be on death row. That is even an ethical battle in our court system today on these details. In essence, should a machine be determining a persons fate, whether life or death or even whether they should receive a loan or not? In our experiments, we will be training them on the admission information for SMUs Data Science Masters program. As this is not determining whether a person would be accepted into the program, we are only answering questions for potential students, this is an ethical application of AI.

6 Analysis Performed and Results

6.1 Methodology

We utilized the domain of admission to the Masters in Data Science program, SMU. To build the necessary chatbots, we employed a data set gathered by another Master's Student group. To gather the data, the group used Counselors and Students as primary sources. They asked SMU Admissions Counselors to provide all the questions they are asked. They sent out surveys to participants of the SMU Master's program to get questions that they had when enrolling to the program. The domain of admission questions has boundaries, thus identifying the intent of each question was not complicated. To make sure that the data would encompass close to all possible questions we create variations for each question because there are any ways to ask the same question.

Next, we built three chatbots, one in IBM Watson, another in Amazon Lex, and another in Microsoft Azure. As we built the chatbots, we noticed the functionality of the back end of the chatbot. IBM Watson and Amazon Lex both have user friendly interfaces that guide a user through the process of creating the decisions of the chatbot whereas Microsoft Azure involves complex coding to create a decision tree.

The common terminology of developing a chatbot in these three AI platforms includes intents and sample utterances. The platforms work off of intents, think goals, of what the user would want.

With the intent identified, we categorized each question into overarching categories. Questions with different intents but with the same overarching topic would be classified as one bucket. We had questions pertaining to Admissions, Campus, Class, Costs and Financials, Professors and Faculty, Program Quality, Student Services, and Technology. For example, the question "What financial help is there to cover tuition?" pertains to "Cost and Financials" and as the intent to find financial aid, whereas the question "How much does 1 unit cost?" pertains to "Cost and Financials" but has the intent to find pricing.

To use the chatbots, we trained the bots to understand the intent of question being asked using sample utterances. For instance, the intent of the question "will it rain tomorrow?" is to want the condition of the weather whereas we would expect the question "what is the current temperature?" to be answered by the temperature. Looking now to the training data set for SMU Masters of Data Science Admissions Office, some questions include What is the cost per credit hour? and What are the typical course requirements per semester? Some of the questions are more complicated such as, How often and by what means is the program reviewed for the quality and real-world applicability of its curriculum? The intents for this training set include professors/academics, costs/financials/-tuition, accreditation, student services, faculty, requirements, technology, application, jobs, and campus life. Due to the large number of different intents and pieces of data admissions counselors face, the size of the training set should be at least 200 different instances to consider when implementing in a chatbot. This process can be done in the console web interface or using a module or package in a software such as Python.

The next step to the training data was to add the question and answer flow suggestions.

We have trained each AI framework with the same data. Because each AI framework understands and aborts data differently, even if each AI framework has been trained with the same data, they can differ on behavior.

To keep everything the same, each chatbot underwent the same series of questions to assess functionality, reliability, usability, efficiency, maintainability, and accessibility. For the testing and comparison of the different frameworks, Python can be utilized to ensure the testing is the same for each bot.

6.2 Uniqueness Between Frameworks

Loading data to the three different platforms exhibit the inherent differences between the platforms. IBM Watson and Amazon Lex both have a user-friendly interface to train the both with the questions and intents whereas Microsoft Azure involved the understanding of a particular syntax.

While loading the sample utterances into Amazon Lex, we noticed some differences between the data quality requirements compared to the other frameworks. Lex will only accept sample utterances that only consist of Unicode char-

acters, spaces, periods (for abbreviations), underscores, apostrophes, and hyphens. IBM Watson and Microsoft Azure had full understanding of punctuation and extra spacing. The stringent requirement for Lex does not tolerate common punctuation such as commas, parentheses, numeric values, and question marks yet it is allowed in the actual testing and use of the chatbot.

Using Python to connect to the platforms and make calls to the chatbots, we immediately can identify the differences in responses. IBM Watson and Microsoft Azure provide a confidence score in the response produced by the chatbot. In comparison, Amazon Lex does not provide a score such as the others; it responses with any messages and best fitting intent among other tags not needed in this analysis. For this reason, we compared the accuracy of each bot by calculating the number of correct intents identified. Even though IBM Watson and Microsoft Azure provide a confidence score, the scale of the scores are different. IBM Watson uses a confidence scoring scale from 0 to 1 while Microsoft Azure uses a scale from 0 to 99.

Amazon Lex can be programmed further to allow for different options, called slot types, when asking for something, called a slot. An easy example of this to understand would be ordering a drink at a coffee shop making the intent along the lines of orderCafeDrink. A sample utterance would be I would like a (BeverageType) where the name BeverageType is the slot with a developer-defined slot type of cafeDrinkType. The values available for the slot type cafeDrinkType could include hot chocolate, coffee, espresso, or tea [4]. This example would be trained to handle text similar to the sample utterance with any of the slot values as the users preference. These slots and slot types allow for developers to use these variables in different utterances without having to explicitly spell out every sample utterance to handle each type of beverage type [4]. However, we did not code any slot or slot types in our chatbot to code the different frameworks similarly to give a better comparison.

6.3 Metrics Results

In order for differences to be identifiable between the three platforms, training each platform with the same training dataset is essential. Below, you can see the first five training questions used to train each platform. We have specified a "Category" and "Intent" for each question.

Table 1. Sample of training questions with broader categories and specific intents.

	Questions	Category	Intent (CLASS_LABELS)
0	What is the average age of students admitted to the program?	Admissions	Admissions_ClassProfile
1	What is the avergae GPA of students admitted to the program?	Admissions	Admissions_ClassProfile
2	Where are the students in the program located?	Admissions	Admissions_ClassProfile
3	Are there any prerequisites?	Admissions	Admissions_Requirements
4	Can I just take classes?	Admissions	Admissions_Requirements

With the use of Python, we test all three platforms with the same use cases. As the simplest response a user can give, we use "hi" as a test case. Even though we did not train any of the bots with the intent of the word "hi", Microsoft Azure knows its intention is "hello" and returns a confidence score of 1 (complete confidence). With the same use case, IBM Watson returns an empty intent and Amazon Lex errors out because its response does not contain an intent at all. This is where we identify another difference. When a platform cannot identify an intent, IBM Watson responds with an empty response, Microsoft Azure returns a message replacing the intent specifying that it cannot identify an intent, and Amazon Lex does not return an intent at all.

The first test set contains four questions that are clearly outside of the domain. It is important to establish limits of your domain. Below, you can see the four questions used.

```
['What is the weather like?',
 'Can I use my refund to buy Hamilton: An American Musical tickets?',
 "Why did SMU's football team get the death penalty?",
 'Is everyone at SMU pretentious?']
```

Fig. 2. Summary of AI Frameworks by Metric

Below, we can see intent results from each platform.

Table 2. Test questions outside of domain from trained questions and intents with results by framework.

Questions	**Watson**	**Lex**	**Azure**
What is the weather like?	Class_Structure	Class_Structure	Class_Structure
Can I use my refund to buy Hamilton: An American Music tickets?	NULL	Admissions_Status	Admissions_Requirements
Why did SMU's football team get the death penalty?	NULL	Class_Structure	Admissions_Requirements
Is everyone at SMU pretentious?	NULL	Class_Structure	Campus_Library

From the table above, we can see that oddly all three platforms associated the question "What is the weather like?" with the intent Class_Structure. IBM Watson was the only platform that did not provide an intent for three of the questions. Microsoft Azure and Amazon Lex returned intents for the last three questions, but the intents did not have anything to do with the question. When evaluating these platforms in this initial test, we can appreciate IBM Watson's null response. IBM Watson's inability to return an intent has more potential than the completely incorrect intents given by Microsoft Azure and Amazon Lex.

Next, we test with our second testing set. This set contains questions that pertain to our domain. First, we want to see what percentage of the questions each platform can return the intent. This second test set contains 47 questions. Below, you can see the first five questions along with the category and intent from our second test set.

Table 3. Sample of test questions with broader categories and specific intents.

	Questions	**Category**	**Intent (CLASS_LABELS)**
1	Who is the professor with most experience?	Professors and Faculty	Faculty_Experience
2	How much does 1 unit cost?	Costs and Financial	Costs_Tuition
3	How many statistic courses are given in the program?	Program Quality	Program_Requirements
4	How many computer science courses are given in the program?	Program Quality	Program_Requirements
5	Are there any courses on engineering?	Program Quality	Program_Requirements

When using this test set, we notice a couple of differences. In ten seconds, IBM Watson returns an intent (either null or a guess). We can easily calculate the number of correct and wrong intents.

From IBM Watson, the proportion of correct responses was 56.5% and the proportion of wrong responses was 43.5%.

In the output above, we can see that about 57% of the questions have a correct intent returned where 43% of the questions do not have a correct intent returned.

When running all 47 questions into Microsoft Azure, we come across an error message stating "Too many requests." If we run the case with resting periods of two minutes, we get further in the intent return process, but eventually we receive the error message again. Thus, we will subset the test set to ten questions. Even with a smaller test set, in order to get intents returned, we have to wait a minute. When Microsoft Azure is done, it returns the follow output.

From Microsoft Azure, the proportion of correct responses was 80% and the proportion of wrong responses was 20%.

Previously mentioned, Amazon Lex does not return anything on intent if it cannot identify the intent. It does not return a null intent or a message in the place of an intent. Because of this difference, is Amazon Lex encounters a question it cannot assign an intent, it will break the loop. Thus, we need to run questions through the chatbot in Amazon Lex that we are sure it can assign some type of intent. Thus, we used the same ten question testing set we used on Microsoft Azure. When Amazon Lex is done, it returns the following output.

From Amazon Lex, the proportion of correct responses was 40% and the proportion of wrong responses was 60%.

When comparing Microsoft Azure and Amazon Lex, we can see that Microsoft Azure has a higher proportion of correct responses.

IBM Watson and Microsoft Azure are the two platforms that provide a confidence score. We want to look at how confident these platforms are when they are assigning an intent. To have results on the same data, we test with a test set of ten questions. Below are the ten questions with the correct intents.

Table 4. Test set of ten questions with correct intents.

ID	Questions	Correct Intent
1	Who is the professor with most experience?	Faculty_Experience
2	How much does 1 unit cost?	Costs_Tuition
3	How many statistic courses are given in the program?	Program_Requirements
4	How many computer science courses are given in the program?	Program_Requirements
5	Are there any courses on engineering?	Program_Requirements
6	When is the FAFSA deadline?	Costs_Financial Aid
7	What is the program accreditation?	Program_Validation
8	Where do I apply for FAFSA?	Costs_Financial Aid
9	What is the average age of students in the program?	Admissions_ClassProfile
10	What is the difference between immersion and capstone?	Program_Requirements

Below are IBM Watson's results to the above questions.

Table 5. Results using the chatbot in IBM Watson for the test set of 10 in Table 4. Results include confidence scores and if the guessed intent was correct by question.

ID	IBM Confidence Score	IBM Intent Guess	IBM Correct Intent
1	0.24	Faculty_Experience	TRUE
2	0.95	Costs_Tuition	TRUE
3	0.40	Program_Requirements	TRUE
4	0.48	Program_Requirements	TRUE
5	0.41	Class_Structure	FALSE
6	0.25	Admissions_Requirements	FALSE
7	0.94	Program_Validation	TRUE
8	0.94	Admissions_Requirements	FALSE
9	0.74	Admissions_ClassProfile	TRUE
10	0.61	Program_Requirements	TRUE

Below are Microsoft Azure's results to the above questions.

Table 6. Results using the chatbot in Microsoft Azure for the test set of 10 in Table 4. Results include confidence scores and if the guessed intent was correct by question.

ID	Azure Confidence Score	Azure Intent Guess	Azure Correct Intent
1	54.28	Faculty_Experience	TRUE
2	22.32	Costs_Tuition	TRUE
3	34.25	Program_Requirements	TRUE
4	27.01	Program_Requirements	TRUE
5	48.24	Program_Requirement	TRUE
6	30.43	Admissions_Requirements	FALSE
7	76.75	Program_Validation	TRUE
8	23.47	Admissions_Requirements	FALSE
9	62.16	Admissions_ClassProfile	TRUE
10	30.33	Program_Requirements	TRUE

To compare the confidence intervals, as shown below in Table 7, the Microsoft Azure confidence interval was normalized to mirror IBM Watson's confidence score scale. The variance in Confidence Scores is the difference between IBM Confidence Score and a normalized Azure Confidence Score.

Table 7. Comparing confidence scores between the chatbots programmed in IBM Watson and Microsoft Azure.

ID	Questions	Variance in Confidence Scores	Description
1	Who is the professor with most experience?	-0.30	Azure is more confident
2	How much does 1 unit cost?	0.73	IBM is more confident
3	How many statistic courses are given in the program?	0.06	IBM is more confident
4	How many computer science courses are given in the program?	0.21	IBM is more confident
5	Are there any courses on engineering?	-0.08	Azure is more confident
6	When is the FAFSA deadline?	-0.06	Azure is more confident
7	What is the program accreditation?	0.17	IBM is more confident
8	Where do I apply for FAFSA?	0.71	IBM is more confident
9	What is the average age of students in the program?	0.11	IBM is more confident
10	What is the difference between immersion and capstone?	0.31	IBM is more confident

For the question "Are there any courses on engineering?", IBM Watson had the incorrect intent whereas Microsoft Azure returned the correct intent. For questions "When is the FAFSA deadline?" and "Where do I apply for FAFSA?", both IBM Watson and Microsoft Azure returned the same incorrect intent. We can see that IBM Watson is more confident in its correct answers than Microsoft Azure is on its correct answers.

Functionality In Section 4, we defined functionality as the measurement of how the chatbot serves its purpose practically.

Practically, each chatbot serves its purpose. It is useful that Microsoft Azure has the automatic knowledge of knowing what the intent of the word "hi" is. From our tests, Azure had the highest proportion of correct intents. IBM Watson is not too far from Azure in proportion of correct intents. Watson was able to run all the questions in the second test set where Azure and Amazon Lex were not.

Reliability Measuring reliability tells us how consistent and dependent the chatbot performs. IBM Watson was the only platform that was consistently returning intents (including null intents) for all 47 questions with no delay. IBM Watson was the platform that was most reliable when testing large data sets. Even with ten questions in the test set, Microsoft Azure would return an error message or had a large run time. Amazon Lex with ten questions in the test set has a large run time compared to IBM Watson.

Usability Usability is defined to measure how applicable and useful the chatbot is. When comparing these three platforms, IBM Watson had a user friendly interface and utilizes relatively simple Python code to return a response with an intent. Microsoft Azure and Amazon Lex required more complex Python code to return a response that was analyzable.

Efficiency Another factor to consider in evaluating chatbots is efficiency in how the chatbot performs in competence and productivity, using computing resources frugally. In these implementations, the training data set had a total of 222 sample utterances spread across 32 intents while the test set had 50 questions to validate the chatbots. Since we were only interested in the bots calculating what the intents were (rather than answering the questions or performing actions of scheduling appointments or ordering goods), these chatbots were not pushed to their fullest extent to test true efficiency and how much the bots could ingest.

However, since we were using modules in Python to code and test the bots, we noticed that Microsoft Azure could only handle 10 observations at a time from the test, and this took about 30 seconds to run. This was an outlier from the other chatbots as both IBM Watson and Amazon Lex could handle the full test set in a timely manner.

Maintainability Maintainability is the measurement of how the device overcomes failures and supports operation and maintenance (O&M) in sustainability of the chatbot.

In the user test case of "hi", we noticed how each platform has a different way of report that it cannot assign an intent. For this reason, IBM Watson returns null for intent so it can easily overcome failures. This feature allows users to identify when IBM Watson cannot identify an intent and possible reassign one manually. Microsoft Azure and Amazon Lex's response eliminate all trace of an intent if it cannot find one.

Accessibility Measuring accessibility tells us how the deployed chatbot is available on other operating systems and devices. We used web user interfaces and Python to train and test our chatbots. The user interfaces for these frameworks allows for ease of sue by guiding the user through building a bot rather than strictly coding it. The Python modules are appropriate for those who appreciate coding. Also, these frameworks allow the functionality to use on many devices such as smartphones. Amazon Lex has an added benefit to incorporate Amazon Alexa services to use their Echo devices with the chatbot.

Metrics Summary The above descriptions have been tallied into three categories to classify if the framework does not meet, meets, or exceeds expectations by metric. See 3.

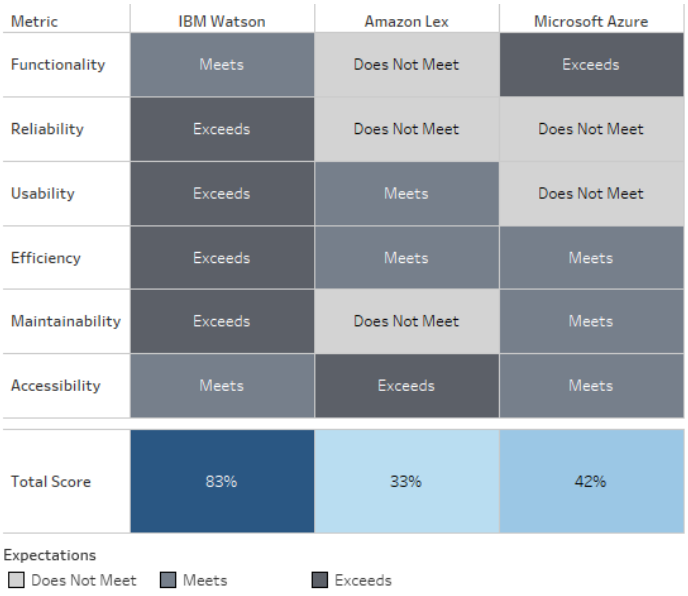

Fig. 3. Summary of AI Frameworks by Metric

The total score was calculated on a one to three scale for each metric based on the expectations categories described above and then finally summed.

7 Conclusions

Artificial Intelligence is quickly gaining traction in businesses, and many are searching for which framework best suits their environment. Six metrics to gauge these frameworks on overall effectiveness included functionality, reliability, usability, efficiency, maintainability, and accessibility. Based on these criteria, IBM Watson is the best AI framework for deploying chatbots.

Acknowledgments. First, Ruby and Crystal would like to thank their faculty mentor, Raghuram Srinivas, for advising them on this project as well as Cory Adams and Kumar Raja Guvindan Raju for collaborating on the data set creation.

Ruby would like to thank Crystal for being a great collaborator and most importantly a great friend.

Ruby is grateful for her mother, who since Rubys birth has encouraged Ruby to learn. Ruby will always be thankful for the endless love and support her mother gives her. Ruby owes everything to that driven, caring, and strong woman she calls her mother.

Ruby is grateful for her sister. A younger sister, but as wise as they come. Ruby has learned so much from her sister and is consistently amazed by her sister. She thanks her sister for the unconditional love and support throughout her life.

Ruby would like to thank her two cousins, Cesar and Christina, for being great role models and for their continuous support.

Ruby wants to extend special thanks to her employer the University of California, Berkeley for supporting her throughout this journey of completing her masters degree.

Crystal would like to thank Ruby for being a wonderful partner in crime and collaborator, especially often in the wee hours.

Crystal is grateful for her proudest supporter and sweet mother, June Todd. Throughout everything, her mom has always been her cheerleader, coach, and best friend. Crystal owes her accomplishments to her mom's good heart, kind soul, and love for math.

Crystal would also like to acknowledge her dearest and longest friend and practically sister, Taylor, for the encouragement and support throughout this process as well as always being in for countless homework dates at Chick-Fil-A.

Crystal wants to extend special thanks to Billy, Amber, and Luke Paul, who she refers to as her second family. She is blessed by them and glad to serve alongside them in ministry.

Crystal is grateful for the Barrus clan of nine, especially Jonah and Charlie. Their family has been a large part of her life for many years, and she is thankful for their friendship.

Crystal would like to thank Mike Stuckey, her coworkers, and Lockheed Martin Corporation for supporting her throughout this journey of completing her masters degree.

Finally, Crystal and Ruby acknowledge God as their Savior and greatest friend in whom she finds strength, which has been especially true during the stresses of graduate school.

References

1. Bradley, T.: Facebook AI Creates Its Own Language In Creepy Preview Of Our Potential Future In: Forbes (2017-07-31) https://www.forbes.com/sites/tonybradley/2017/07/31/facebook-ai-creates-its-own-language-in-creepy-preview-of-our-potential-future/#73ed5678292c
2. Chang, M., Cho, H., Wu, D.: Legal Issues of Siri and AI Systems https://sites.google.com/site/cs181siri/false-advertisement-and-performance-metrics/b-metrics-of-performance
3. Clauser, G.: What Is Alexa? What Is the Amazon Echo, and Should You Get One? In: The Wirecutter (2017-09-05) http://thewirecutter.com/reviews/what-is-alexa-what-is-the-amazon-echo-and-should-you-get-one/
4. Hira, N., Pimpalkhute, H.: Building Better Bots Using Amazon Lex (Part 1) In: AWS AI Blog (2017-02-24) https://aws.amazon.com/blogs/ai/building-better-bots/
5. Lovejoy, B.: Marriott testing Siri and Alexa to decide which will control devices in its hotel rooms In: 9To4Mac (2017-03-22) https://9to5mac.com/2017/03/22/marriott-aloft-siri-alexa/
6. Rhinehart, C.: 10 Things You Need to Know About the Technology Behind Watson In: Entrepreneurial and Intrapreneurial Insights (2011-01-17, Retrieved 2016-01-10)

A Appendix

```
import pandas as pd
import numpy as np

#hides warning outputs
import warnings
warnings.filterwarnings('ignore')

#Read in data
questions = pd.read_csv('Questions - Category - Intent.csv')
questions.head()

questions.info()

questions.describe()
test_the_train= questions.take(np.random.permutation(len(
    questions))[:10])
```

```
test_the_train

#We are testing only 10 Questions to see if Chat bots were
    trained well
df_text=list(test_the_train["Questions"])
df_text

#Watson - Trained by our data
from watson_developer_cloud import ConversationV1
import json

for index in range(len(df_text)):

conversation =ConversationV1(
username = 'Insert Username',
password = 'Insert Password',
version = '2017-05-26'
)


context = {}

workspace_id = '7fe4c347-934d-436a-a2a0-a094f5ea85f7'

response = conversation.message(
workspace_id = workspace_id,
message_input = {'text': df_text[index] },
context = context
)

raw = (json.dumps(response, indent=2))
raw_dict = json.loads(raw)
print raw_dict['intents']


#Microsoft Azure - trained by our data
import json
import urllib2

for index in range(len(df_text)):
data = {
"question": df_text[index]
}

req = urllib2.Request('https://westus.api.cognitive.microsoft
    .com/qnamaker/v1.0/knowledgebases/900d4e99-fb2b-49ee-93d1
    -7dce4deb1b89/generateAnswer')
req.add_header('Ocp-Apim-Subscription-Key', 'Insert Key')
req.add_header('Content-Type', 'application/json')
req.add_header('Cache-Control', 'no-cache')
```

```
response = urllib2.urlopen(req, json.dumps(data))
respString = response.read()
print json.loads(respString)

#Amazon Lex - Trained by Our Data
## install package boto3
##!pip install boto3
## load package boto3
import boto3
## aws lex access keys
AWSAccessKeyId = "Insert key" #Access Key
AWSSecretKey = "Insert Secret key" #Secret Key

## initialize connection to lex
client = boto3.client('lex-runtime',
region_name = 'us-east-1',
aws_access_key_id = AWSAccessKeyId,
aws_secret_access_key = AWSSecretKey)

## Need questions and corrent intents from test set
stringy = list(test_the_train['Questions'])
intent = list(test_the_train['intent (CLASS_LABELS)'])

## Number of questions in test set
n = test_the_train.count(axis = 0)['Questions']

## Initialize variables to calculate proportions of correct
    responses from Lex
countCorrect = 0
countWrong = 0

print intent

for i in range(len(test_the_train)):
text = stringy[i]
answer = intent[i]
response = client.post_text(botName = 'SMUAdmissionBot',
botAlias = 'Test',
userId = 'DreamTeamUser',
inputText = text)['intentName']

if response == answer:
countCorrect = countCorrect + 1
else:
countWrong = countWrong + 1

print response
```

```
print("Proportion of Correct Responses from Lex: ",
    countCorrect / (len(test_the_train) * 1.0))
print("Proportion of Wrong Responses from Lex: ", countWrong
    / (len(test_the_train) * 1.0))


#Testing Empty Questions - IBM Watson

conversation =ConversationV1(
username = 'Insert Username',
password = 'Insert Password',
version = '2017-05-26'
)


context = {}

workspace_id = '7fe4c347-934d-436a-a2a0-a094f5ea85f7'

response = conversation.message(
workspace_id = workspace_id,
message_input = {'text': '' },
context = context
)

raw = (json.dumps(response, indent=2))
raw_dict = json.loads(raw)
print raw_dict['intents']


#Testing Empty Questions - Microsoft Azure
data = {
"question": ''
}

req = urllib2.Request('https://westus.api.cognitive.microsoft
    .com/qnamaker/v1.0/knowledgebases/900d4e99-fb2b-49ee-93d1
    -7dce4deb1b89/generateAnswer')
req.add_header('Ocp-Apim-Subscription-Key', 'Insert Key')
req.add_header('Content-Type', 'application/json')
req.add_header('Cache-Control', 'no-cache')

response = urllib2.urlopen(req, json.dumps(data))
respString = response.read()
print json.loads(respString)

#Testing Empty Questions - Amazon Lex
text = ''
response = client.post_text(botName = 'SMUAdmissionBot',
botAlias = 'Test',
```

```
userId = 'DreamTeamUser',
inputText = text)['intentName']
print response

#Testing one word out of context - IBM Watson
conversation = ConversationV1(
username = 'Insert Username',
password = 'Insert Password',
version = '2017-05-26'
)


context = {}

workspace_id = '7fe4c347-934d-436a-a2a0-a094f5ea85f7'

response = conversation.message(
workspace_id = workspace_id,
message_input = {'text': 'hi' },
context = context
)

raw = (json.dumps(response, indent=2))
raw_dict = json.loads(raw)
print raw_dict['intents']

conversation =ConversationV1(
username = 'Insert Username',
password = 'Insert Password',
version = '2017-05-26'
)


context = {}

workspace_id = '7fe4c347-934d-436a-a2a0-a094f5ea85f7'

response = conversation.message(
workspace_id = workspace_id,
message_input = {'text': 'ruby' },
context = context
)

raw = (json.dumps(response, indent=2))
raw_dict = json.loads(raw)
print raw_dict['intents']


#Testing one word out of context - Mircrosft Azure
data = {
```

```
"question": 'hi'
}

req = urllib2.Request('https://westus.api.cognitive.microsoft
    .com/qnamaker/v1.0/knowledgebases/900d4e99-fb2b-49ee-93d1
    -7dce4deb1b89/generateAnswer')
req.add_header('Ocp-Apim-Subscription-Key', 'Insert Key')
req.add_header('Content-Type', 'application/json')
req.add_header('Cache-Control', 'no-cache')

response = urllib2.urlopen(req, json.dumps(data))
respString = response.read()
print json.loads(respString)

data = {
"question": 'ruby'
}

req = urllib2.Request('https://westus.api.cognitive.microsoft
    .com/qnamaker/v1.0/knowledgebases/900d4e99-fb2b-49ee-93d1
    -7dce4deb1b89/generateAnswer')
req.add_header('Ocp-Apim-Subscription-Key', 'Insert Key')
req.add_header('Content-Type', 'application/json')
req.add_header('Cache-Control', 'no-cache')

response = urllib2.urlopen(req, json.dumps(data))
respString = response.read()
print json.loads(respString)


#Testing one word out of context - Amazon Lex
text = 'hi'
response = client.post_text(botName = 'SMUAdmissionBot',
botAlias = 'Test',
userId = 'DreamTeamUser',
inputText = text)

if response['message']== 'Sorry, can you please repeat that?'
    :
print "No Guess"
else: print response['intentName']

text = 'ruby'
response = client.post_text(botName = 'SMUAdmissionBot',
botAlias = 'Test',
userId = 'DreamTeamUser',
inputText = text)

if response['message']== 'Sorry, can you please repeat that?'
    :
```

```
print "No Guess"
else: print response['intentName']


#Testing
#Read in data
test = pd.read_csv('Capstone Test Set.csv')
test.head()

ridic = test.loc[test["Category"]=="Ridiculous",:]
ridic_test = list(ridic["Questions"])
ridic_test

norm = test.loc[test["Category"]!="Ridiculous",:]
norm_test = list(norm["Questions"])
norm_test_answers = list(norm['intent (CLASS_LABELS)'])

norm.head()

#IBM Watson

from watson_developer_cloud import ConversationV1
import json

for index in range(len(ridic_test)):

conversation =ConversationV1(
username = 'Insert Username',
password = 'Insert Password',
version = '2017-05-26'
)


context = {}

workspace_id = '7fe4c347-934d-436a-a2a0-a094f5ea85f7'

response = conversation.message(
workspace_id = workspace_id,
message_input = {'text': ridic_test[index] },
context = context
)

raw = (json.dumps(response, indent=2))
raw_dict = json.loads(raw)
print raw_dict['intents']

#Microsoft Azure
import json
import urllib2
```

```
for index in range(len(ridic_test)):
data = {
"question": ridic_test[index]
}

req = urllib2.Request('https://westus.api.cognitive.microsoft
    .com/qnamaker/v1.0/knowledgebases/900d4e99-fb2b-49ee-93d1
    -7dce4deb1b89/generateAnswer')
req.add_header('Ocp-Apim-Subscription-Key', 'Insert Key')
req.add_header('Content-Type', 'application/json')
req.add_header('Cache-Control', 'no-cache')

response = urllib2.urlopen(req, json.dumps(data))
respString = response.read()
print json.loads(respString)
print ridic_test[index]


#Amazon Lex
for i in range(len(ridic_test)):
text = ridic_test[i]
response = client.post_text(botName = 'SMUAdmissionBot',
botAlias = 'Test',
userId = 'DreamTeamUser',
inputText = text)['intentName']

print response

#Watson
from watson_developer_cloud import ConversationV1
import json

## Initialize variables to calculate proportions of correct
    responses from IBM
countCorrect = 0
countWrong = 0

for index in range(len(norm_test)):


conversation =ConversationV1(
username = 'Insert Username',
password = 'Insert Password',
version = '2017-05-26'
)


context = {}
```

```
workspace_id = '7fe4c347-934d-436a-a2a0-a094f5ea85f7'

response = conversation.message(
workspace_id = workspace_id,
message_input = {'text': norm_test[index] },
context = context
)

raw = (json.dumps(response, indent=2))

raw_dict = json.loads(raw)

resp = raw_dict["intents"][0]["intent"]
answer = norm_test_answers[index]

if resp == answer:
countCorrect = countCorrect + 1

else:
countWrong = countWrong + 1

#print raw_dict['intents']

##print("Proportion of Correct Responses from Lex: ",
    countCorrect / (n * 1.0))
##print("Proportion of Wrong Responses from Lex: ",
    countWrong / (n * 1.0))
print("Proportion of Correct Responses from IBM: ",
    countCorrect / (len(norm_test) * 1.0))
print("Proportion of Wrong Responses from IBM: ", countWrong
    / (len(norm_test) * 1.0))

#Microsoft Azure
## Initialize variables to calculate proportions of correct
    responses from Azure
countCorrect = 0
countWrong = 0

import json
import urllib2

for index in range(10):
data = {
"question": norm_test[index]
}

req = urllib2.Request('https://westus.api.cognitive.microsoft
    .com/qnamaker/v1.0/knowledgebases/900d4e99-fb2b-49ee-93d1
    -7dce4deb1b89/generateAnswer')
req.add_header('Ocp-Apim-Subscription-Key', 'Insert Key')
```

```
req.add_header('Content-Type', 'application/json')
req.add_header('Cache-Control', 'no-cache')

response = urllib2.urlopen(req, json.dumps(data))
respString = response.read()
resp_dict = json.loads(respString)

resp = resp_dict['answer']
answer = norm_test_answers[index]

if resp == answer:
countCorrect = countCorrect + 1

else:
countWrong = countWrong + 1

print("Proportion of Correct Responses from Azure: ",
    countCorrect / (10*1.0) )
print("Proportion of Wrong Responses from Azure: ",
    countWrong / (10*1.0) )


#Amazon Lex
## Initialize variables to calculate proportions of correct
    responses from Lex
countCorrect = 0
countWrong = 0
for i in range(10):
text = norm_test[i]
answer = norm_test_answers[i]
response = client.post_text(botName = 'SMUAdmissionBot',
botAlias = 'Test',
userId = 'DreamTeamUser',
inputText = text)['intentName']
if response == answer:
countCorrect = countCorrect + 1
else:
countWrong = countWrong + 1

print response

print("Proportion of Correct Responses from Lex: ",
    countCorrect / (10*1.0) )
print("Proportion of Wrong Responses from Lex: ", countWrong
    / (10*1.0) )

#Comparing IBM and Azure confidence intervals
## Initialize variables to calculate proportions of correct
    responses from IBM
countCorrect = 0
```

```
countWrong = 0

for index in range(10):
conversation = ConversationV1(
username = 'Insert Username',
password = 'Insert Password',
version = '2017-05-26'
)


context = {}

workspace_id = '7fe4c347-934d-436a-a2a0-a094f5ea85f7'

response = conversation.message(
workspace_id = workspace_id,
message_input = {'text': norm_test[index] },
context = context
)

raw = (json.dumps(response, indent=2))

raw_dict = json.loads(raw)

resp = raw_dict["intents"][0]["intent"]
answer = norm_test_answers[index]

print raw_dict['intents']
print norm_test[index]

for index in range(10):
data = {
"question": norm_test[index]
}

req = urllib2.Request('https://westus.api.cognitive.microsoft
    .com/qnamaker/v1.0/knowledgebases/900d4e99-fb2b-49ee-93d1
    -7dce4deb1b89/generateAnswer')
req.add_header('Ocp-Apim-Subscription-Key', 'Insert Key')
req.add_header('Content-Type', 'application/json')
req.add_header('Cache-Control', 'no-cache')

response = urllib2.urlopen(req, json.dumps(data))
respString = response.read()
resp_dict = json.loads(respString)

resp2 = resp_dict
answer2 = norm_test_answers[index]

print answer2
```

```
print resp2
```