Utilizing Bots in Delivering Content from Kentico Cloud and Kentico EMS

Antti Eikonsalo

University of Tampere
Faculty of Natural Sciences
Degree programme in Computer Science
M. Sc. thesis
Supervisor: Erkki Mäkinen
October 2017

University of Tampere
Faculty of Natural Sciences
Degree Programme in Computer Science
Antti Eikonsalo: Utilizing Bots in Delivering Content from Kentico Cloud and Kentico EMS
M. Sc. thesis, 49 + 1 attachment pages
October 2017

This thesis explores an interesting and current topic of utilizing chatbots in content delivery. It defines the important terms and concepts for approaching the subject and offers a brief history of chatbots to provide some background on the topic.

As the thesis is written for the software company Kentico, it specifically focuses on delivering content from Kentico Cloud and Kentico EMS software. Thus, the main research question the thesis aims to answer is what are the options for utilizing chatbots in delivering content from Kentico software. In order to achieve this goal, the thesis introduces the Kentico software and the available options, and then moves on to analyze their compatibility.

As a result of the analysis, the thesis concludes that all of the considered options would be suitable for delivering content from the Kentico software. However, the extent of their suitability depends on what the user is looking for. While some options are better suited for complex implementations, they would also require more effort to develop. On the other hand, some of the options that offer more limited features do not require such wide technical knowledge and are faster to develop and set up.

Keywords: Dialog system, chatbot, content delivery, Kentico

Contents

1. Introduction

While chatbots and the basic technology they use have been around for a significant amount of time, their use has become more and more common recently. This can be seen, for example, in the rise in popularity of personal assistants, such as Siri on the iOS and OS X platforms, Cortana on Windows, and Google Now on Android. Also, products such as Google Home, Amazon Alexa, and many other services have become more common lately [PR Newswire 2014]. Even before this, a type of a chatbot has been used, and is still used, for a long time in some companies' phone services, where the caller is routed to the correct line by an automated system, which takes input either as keypad presses or by voice detection [Berg 2010; Wikipedia$_b$]. While these systems may have used key-based navigation, some also supported voice control to navigate the menus.

The goal of the thesis is to identify the plausible chatbot solutions that could be leveraged in delivering content from Kentico Cloud and Kentico EMS platforms. To identify the advantages and the disadvantages of the options, they are analyzed by general principles based on criteria used when selecting third-party libraries internally at Kentico. The thesis was ordered by software company Kentico and was partly written while working at the company in question in Brno, Czech Republic.

The thesis will state that while there are differences between the bot platforms and frameworks, they can all be used to deliver content from Kentico EMS and Kentico Cloud. The thesis shows that the extent to which this is possible depends on both the features offered by the bot platform or framework as well as on the requirements and the implementation of the bot. From Kentico Cloud's point of view, the necessary technical requirements are fulfilled. For Kentico EMS, the REST API offered suffices for content delivery and via customization a custom API endpoint can be defined for any further need.

In Chapter 2, the basic terms and concepts necessary to define will be introduced. Also, the concept of a chatbot will be determined, the different ways in which chatbots can be categorized will be explored, and the components used in chatbots will be described. Additionally, to offer some background on the technological progression of chatbots, an overview of the history of bots will be provided.

The third chapter of the thesis will shortly introduce the Kentico Cloud and Kentico EMS. Also the technical requirements and limitations regarding the two softwares will be discussed in this chapter.

Chapter 4 will describe the different available solutions for creating chatbots. The chapter is divided to two subchapters, platforms and frameworks, which both detail several different options for creating bots that can be used for content delivery.

In the fifth chapter, the available solutions will be analyzed in regard to delivering content from Kentico Cloud and Kentico EMS. The chapter is similarly divided in to two subchapters as the fourth chapter. In the subchapters, the suitability of each available option will be assessed. Finally, in the sixth chapter, the results of the analysis are presented.

2. Terms and Basic Concepts

There are many terms used when talking about chatbots, for example, (human-computer) dialog systems, (conversational) agents, chatterbots, or simply bots. These terms are often used interchangeably with each other, however, distinctions between the terms exist. In order to efficiently analyze the available solutions for delivering content from Kentico EMS and Kentico Cloud, it is necessary to understand what exactly each of these terms mean. Additionally, the concepts closely related to them, such as intent, entity, and response, will be defined in the following subchapter.

2.1. Intent, Entity, and Response

Intent represents the intention of the user, for example, Dialogflow defines it as "the mapping between what the user says and what action should be taken" [Dialogflow$_f$]. Similarly, Saunders [2017] determines intent "as the question or request that a user inputs". An example of this could be, for instance, booking a ticket.

Entities Saunders [2017] describes as "parameters we can pull out of a conversation". Dialogflow goes on to specify that entities represent the detailed information that clarifies the purpose of the intent [Dialogflow$_d$]. To expand on the example offered above, the entity could be a certain movie, that is, the user's intent is to book a ticket for a movie. Additional entities could be, for example, time and location of the movie theater.

Finally, response is the action that is taken as a result of identifying the intent and the entity. The response could be, for example, the data returned for the end user, some actionable data that is returned by the platform or framework, or a call to an external service to perform some action or to fetch external data.

2.2. Dialog Systems

Dialog system is defined by Klüwer [2011, 22] as a "system which can conduct a conversation with another agent, for example a human or another dialog system. Dialog systems can use various forms of input and output (e.g. speech, gestures or text)." Similarly, Berg [2014, 37] describes a dialog system as "a machine that verbally communicates with a human user to exchange information about a common topic". However, Klüwer defines dialog system to be a more advanced system utilizing Natural Language Understanding (NLU) and more complex techniques. She explains that these systems include additional modules to make the system more intelligent. Klüwer goes on to describe it as "a system which can conduct a conversation with another agent, usually a human". According to her, they usually include at least the following additional components compared to traditional chatbots[1]: an input processing and NLU functionality, one component for input management, and NLU features, dialog flow management and an output generation component. [Klüwer, 2011, 3–4]

The definitions of Klüwer and Berg differ in the specificity of the definition. Klüwer describes the dialog system as a more advanced system, which incorporates natural language understanding as well as generating the output instead of simple pattern matching. Berg on the other hand doesn't distinguish between the simple pattern matching and more advanced systems. He defines the term quite vaguely covering systems which offer varying levels of features and components.

Berg and Düsterhöft categorize dialogs based on the initiative. They define initiative as "the dialog partner who has influence on the control of the dialog". Berg and Düsterhöft state that depending on the party that has the control, the dialog systems can be divided into three categories: system-initiative, mixed-initiative, and user-initiative. They determine that a system-initiative system would not allow the user to freely use natural language. Instead, only a simple question-answer dialog would be possible. Additionally, a mixed-initiative system would allow the user to affect the flow of conversation and the system would adapt to user. Furthermore, a user-initiative system expects the user to be the active party and does not attempt to lead the user towards their goal. [Berg and Düsterhöft 2010]

2.3. Conversational Agent and Chatbot

Klüwer [2011, 2] defines chatbots as "text-based interfaces and a stimulus-response pattern-matching algorithm constituting the basis for dialog functionality". Alternatively, she offers a definition: "a text-based system which incorporates pattern-matching methods to conduct a conversation with another agent" [Klüwer 2011, 22]. Along the same lines, AbuShawar and Atwell [2016, 374] determine that chatbot is "a conversational agent that interacts with user's turn by turn using natural language". Furthermore, Abdul-Kader and Woods [2015, 72] describe chatbot as "a computer program that have the ability to hold a conversation with human using Natural Language Speech". Abdul-Kader and

[1] This will be returned to in the next subchapter.

Woods' definition, however, assumes the mode of communication and expects that the bot has the ability to hold a conversation in natural language. This definition is not quite clear as one can assume the bot has an ability to understand the context and the topic of the conversation and produce output to further the conversation as the active party. It is not certain if a pattern matching bot that can match a topic keyword in the input and provide a question on the topic to the user is considered as holding a conversation. In contrast, both Klüwer's and AbuShawar and Atwell's definitions are quite close to each other. They are more clearly defined and they refer to a simpler bot, which utilizes methods such as pattern matching to interact with another agent. While Klüwer's definition includes the ability to conduct a conversation with another agent, it clearly states that the system utilizes pattern matching methods.

As can be seen, the definitions of the terms vary slightly. In the scope of this thesis, a chatbot is defined as a type of dialog system. Additionally, conversational agent and chatbot are understood as synonyms. The thesis will rely on the definition and categorization of dialog systems provided by Berg. By this definition, a dialog system does not include any advanced components such as natural language understanding, natural language generation (NLG), or machine learning (ML).

To provide the functionality the bots offer, there are several related methods that are used. The simplest bots use pattern matching algorithms to identify certain keywords from the user input. The keywords are matched with a template that is the best match and the predefined response is returned as output to the user. However, there are different techniques used to make the bots seem more intelligent than they in reality are. Subchapter 2.5 will explain in more detail what techniques are used by the pattern matching bots, as well as the components used in more advanced bots and what functionality each component entails.

2.4. Categorizing Bots

While there are many ways to categorize bots, in the thesis categorizing of bots is based on the way that the dialog control is handled. According to Berg, dialog systems can be divided to four main categories based on the way the dialogue manager handles the conversation flow: finite-state-based, frame-based, plan-based, and information-stat-based approaches. Berg states that the simplest of these is the finite-state-based approach where the conversation flow is composed of a graph and the conversation advances through the graph by finite-state-automation. The questions asked by the system are represented as the nodes and the user responses are in between the nodes, as can be seen in Figure 1. [Berg 2014, 41] These types of dialogs are inflexible and only allow for simple, predefined order of questions.

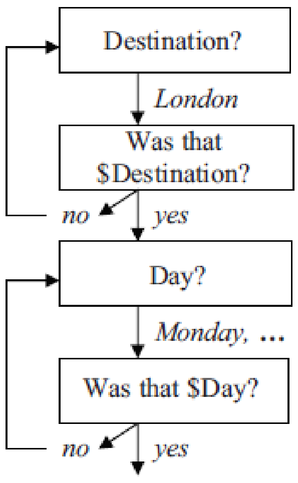


Figure 1: Finite-state graph [Jokinen & McTear 2009, 25].

Frame-based approach can be understood through the analogy of printed forms. In frame-based approach the frames and slots are equivalent to forms and fields, respectively. Each slot in a frame represents an answer to a question (Figure 2). The advantage of this approach is that several slots can be filled from a single user utterance in any order. This allows the user to use a more natural way of communicating by deciding what information is given at what time. [Berg 2014, 42]

$$\begin{bmatrix} destination: & London \\ day: & Monday \\ persons: & n/a \\ start\_date: & n/a \\ end\_date: & n/a \end{bmatrix}$$

Figure 2: Frame [Berg 2014, 42].

Plan-based approach relies on the notion that users of a dialog system are trying to reach some goal. The goal can be represented as a plan consisting of certain steps, which may or may not be dependent on each other, to be taken in order to achieve the goal. Berg uses an example from Jurafsky and Martin (see Figure 3) of an agent A booking a flight F for client C: "The agent has a set of true beliefs (he knows facts) and the client has the desire to book the flight. By making a reservation, the flight is booked and C's desire is accomplished" [Berg 2014, 43].

BOOK-FLIGHT	**(A,C,F)**
Constraints:	$Agent(A) \wedge Flight(F) \wedge Client(C)$
Precondition:	$Know(A, depart-date(F)) \wedge Know(A, depart-time(F)) \wedge$
	$Know(A, origin(F)) \wedge Know(A, flight-type(F)) \wedge$
	$Know(A, destination(F)) \wedge Has-Seats(F) \wedge$
	$Want(C, BOOK(A, C, F)) \wedge \ldots$
Effect:	$Flight-Booked(A, C, F)$
Body:	$Make-Reservation(A, F, C)$

Figure 3: Action scheme [Jurafsky & Martin 2000].

Information-state-approach combines several different strategies, which allows for a wider flexibility. The approach incorporates several components as listed by Berg [2014, 43] (for more detailed descriptions, see Traum and Larsson [2003]):

- Informational components: participants, user models, beliefs, intentions, etc.
- Formal representations of the informational components as lists, typed feature structures, etc.
- Dialogue moves that trigger the update of the information state (including NLU and NLG)
- Update rules that govern the updating of the information state and select a particular dialogue move to perform given conditions on the current information state
- Update strategy for deciding which rules to apply.

The process for developing an informational state system starts with the decisions on what information is to be modeled and how to model it. This produces the data structure that is called the informational state. According to Berg [2014, 44], this can be thought of as a very complex frame, which holds a lot more information than the frames in the frame-based approach.

Berg also states that dialogue moves are used as an abstraction to handle a vast number of possible natural language messages that can be matched to a certain update. For each type of update there must exist at least one dialogue move. He goes on to explain, there are usually two types of moves: ask and answer. Berg explains that update rules formalize the way of updating the information state by defining the conditions when the update is done, the effects of the update, and the selection of the next move. He also mentions that there must also be a strategy based on which the rules are applied. [Berg 2014, 44]

2.5. Components

Probably one of the most common component mentioned alongside chatbots is natural language understanding, which is the process of extracting the meaning of the user input. According to Klüwer, part-of-speech tagging (POS tagging) is used to identify the lexical information of the tokens in the user input. She explains that part-of-speech tagging can consist of, for example, identifying the word class, such as a noun, a verb, or an adjective, inflections, and lemma of words found in the input. Klüwer states that named entity recognition (NER) is like POS tagging in a sense since it identifies and labels the words or sentences that refer to named entities such as people, brands, or company names. For the dialog system to understand the user input also syntactic and semantic parsing is carried out. According to Klüwer, this means detection of higher structural units which are analyzed to provide syntax description: "[…] the grammar-like description of the structure in the sentence, describing which groups of words go together (as constituents) and which words are the subject or object of a verb." Additionally, the semantic analysis will provide the meaning of the sentences. [Klüwer 2011, 8–9] Through the semantic analysis the bot can provide the user with more context aware responses as it will be able to distinguish the subtler aspects of the user's utterances.

According to Klüwer, dialog systems also generally include some output generation module which constructs the output to be presented back to the user. She explains that the module can generate several varying surface strings based on the stored answer's linguistic description while making sure the output is appropriate in regard to the input's structure. This is achieved with the help of external knowledge bases and context gathered from the preceding dialog. [Klüwer 2011, 9] The simplest method to provide output to the user would be to return a pre-stored response from a database. To add some flexibility to the answers, another possibility is to use templates with stored text, which include some variables that can be dynamically replaced. Finally, the most complex option for generating the response is when there is no predefined response and the entire response is generated from scratch.

This process is described by Berg who divides it into three main stages: document planning, microplanning and surface realization. He continues that each one of these stages includes several subtasks. In the document planning phase, the information that will be included is decided as well as the overall structure of the text. Microplanning stage includes the lexicalization, merging of similar sentences to make the response more natural, and choosing which referring expressions, such as personal pronouns instead of a name, and the tense or mood to be used. Finally, in the surface realization stage, the actual text of the response will be generated "by applying grammar rules, inflecting verbs, adding auxiliary verbs and building the correct tense". [Berg 2014, 127]

Different forms of Machine Learning can be used also in bots, for example, to understand the user in a more natural way and to generate more personalized responses [Bertolucci 2016]. For example, a machine learning process called active learning is described next. This process is used, for example, by Microsoft Language Understanding Intelligent Service (LUIS). Microsoft LUIS' documentation describes the machine learning as an iterative process, which can begin once the necessary intent and entities for the bot's domain have been defined. It starts with the initial training of the bot to give it example utterances for each of the intents defined for the bot. This way the bot will learn several utterances it can compare untrained utterances to when it encounters them. After the initial training, the active learning part of machine learning can start. Next the bot is exposed to a number of utterances that it has not been trained to understand. This could be done by, for example, in Microsoft LUIS' case by publishing the bot and allowing a number of real people to interact with the bot. Once the bot has encountered some utterances where it had troubles identifying the intents with a good probability or identified them wrong, the next step of the process can begin. In this stage the bot is taught by a human the correct intents for the misidentified or uncertain intents. [Hanna and Mak 2017] Additionally, neural networks can be utilized, however, the area is too wide to be reviewed in the scope of this thesis.[2]

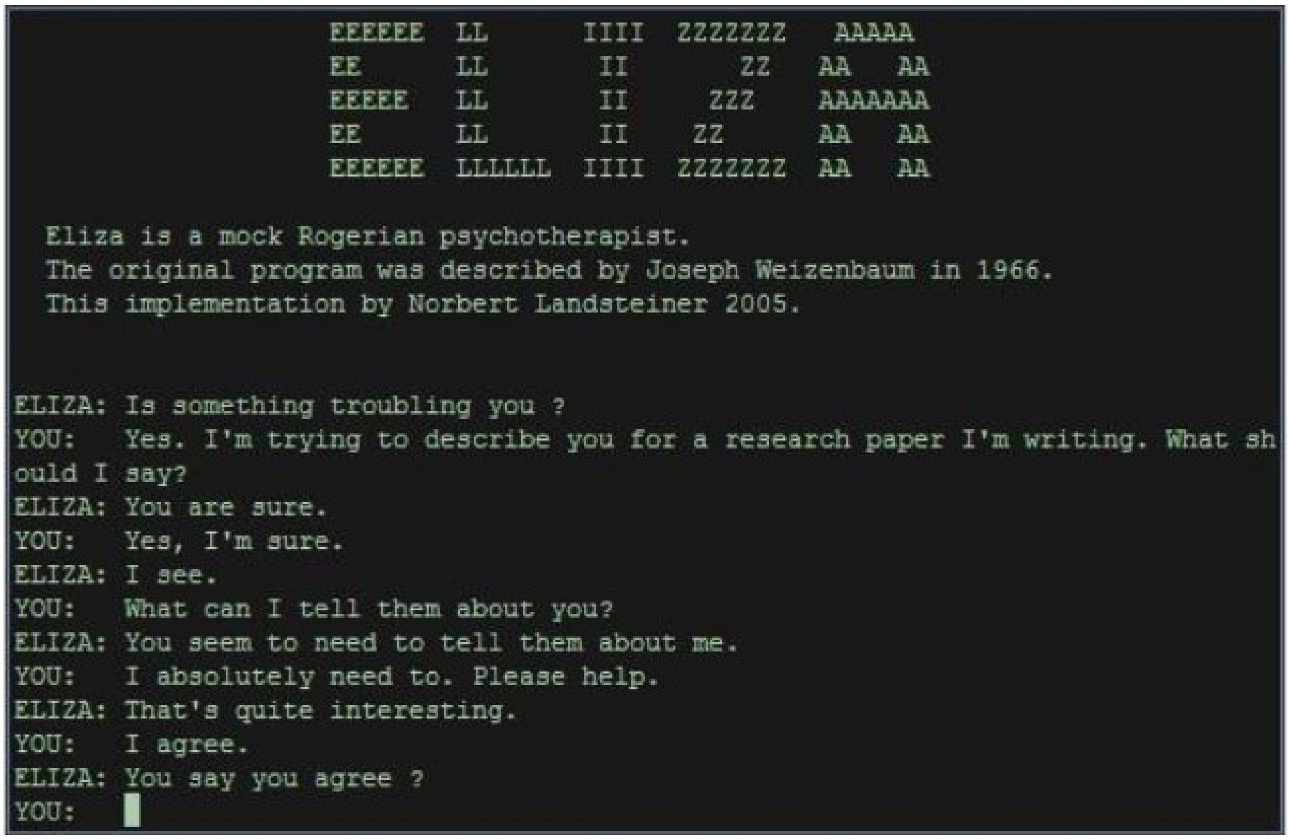
As the relevant terms have been described we can move on to a short overview of how the bots have advanced from the earliest chatbots to more recent ones. In the next subchapter, we will take a look at the chatbot history through some examples.

2.6. Overview of Chatbot History

The history of chatbots is generally considered to have begun in the 1960's when one of the earliest successful chatbots, ELIZA (see Figure 4), was developed by Weizenbaum [1966]. He describes the bot to use simple pattern matching and keyword recognition to match the utterance of the user to a response. However, for the bot to be "intelligent", it requires a large database of prede-

[2] For more information on neural networks, see, for example, Abadi et al. [2016], Sordoni et al. [2015], and Vinyals and Le [2015].

fined patterns to match the user input against. To overcome this limitation Weizenbaum used regular expression matching to match several utterances to one response to narrow down the amount of stored responses in the system. This allows for writing a general pattern for user utterances in which there are countless possible variations. Additionally, he introduced a ranking system for keywords[3]. [Weizenbaum 1966, 36–43]



```
                    EEEEEE  LL      IIII  ZZZZZZZ   AAAAA
                    EE      LL       II       ZZ   AA   AA
                    EEEEE   LL       II     ZZZ    AAAAAAA
                    EE      LL       II    ZZ      AA   AA
                    EEEEEE  LLLLLL  IIII  ZZZZZZZ  AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Yes. I'm trying to describe you for a research paper I'm writing. What sh
ould I say?
ELIZA: You are sure.
YOU:   Yes, I'm sure.
ELIZA: I see.
YOU:   What can I tell them about you?
ELIZA: You seem to need to tell them about me.
YOU:   I absolutely need to. Please help.
ELIZA: That's quite interesting.
YOU:   I agree.
ELIZA: You say you agree ?
YOU:   
```

Figure 4: Sample conversation with an implementation of ELIZA [Radziwill and Benton 2017].

Another problem arises with the rules being too permissive. To illustrate the problem, Klüwer [2011, 4] uses a situation where the chatbot is told the user's name and the bot responds with a greeting which repeats the name user gave:

"User: My name is James and I will wear you down!

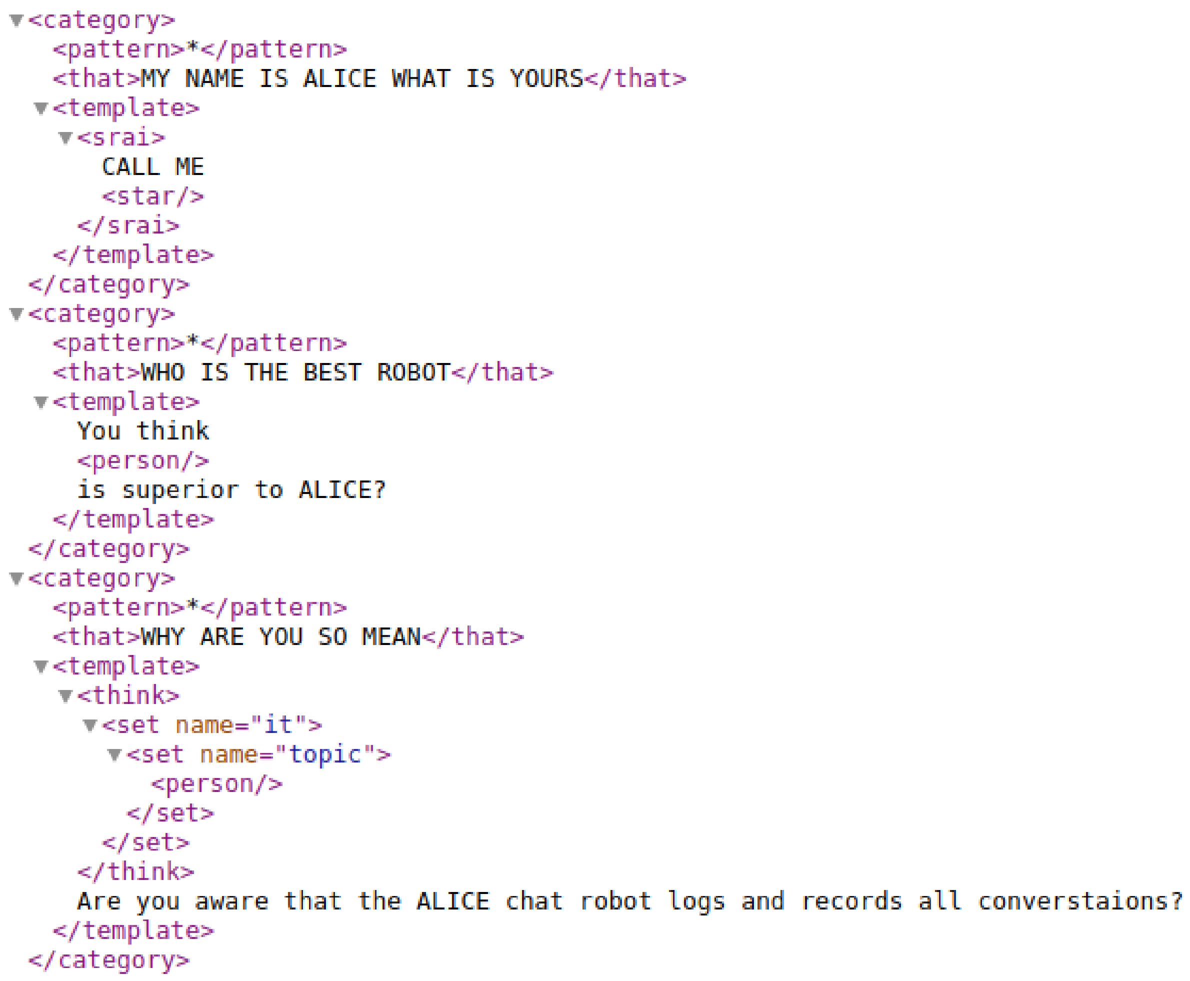
Chatbot: It's a pleasure to meet you James and I will wear you down!"

This is an example that nicely exemplifies the limitations of pattern matching architecture. Klüwer further explains, that to avoid repetition in the answers provided by the bot, it is possible to write multiple reassembly rules for

[3] The keyword ranking system will be described more later in this subchapter.

one decomposition rule. This way the bot's responses seem more natural as they are used in sequential order to avoid repeating the same response. [Klüwer 2011, 4] To optimize the performance of ELIZA, Weizenbaum [1966] introduced the possibility of including keywords and ranking values for keywords in the rules. This allows only the rules that include the keyword to be processed for the input reducing the number of rules to process. He describes the keyword ranking to work in a way that the decomposition rules that include keywords with a higher rank will take precedence in the processing order of rules. Additionally, rules that don't include the keywords will not be processed. [Weizenbaum 1966, 39–42]

Another similar bot based on same concepts as ELIZA is ALICE. It uses the open source Artificial Intelligence Markup Language (AIML), an XML dialect, seen in Figure 5, and is categorized as a stimulus-response bot using pattern matching [Klüwer 2011, 6]. As Klüwer [2011, 6] and Abdul-Kader and Woods [2015, 73] mention, these kinds of bots are as intelligent as the knowledge base backing them is vast. The conversation will quite quickly become repetitive and boring as the user has drained the existing pattern-template pairs in the bot's database. Nevertheless, ALICE's developers have managed to make the conversation more flexible with features such as "[…] recursion, topic annotation, variables and a short-term memory" to allow the bot to have different dialog states [Klüwer, 2011, 8]. ALICE can, for example, remember the previous topic and its own previous reply. This allows for the bot to respond in a more meaningful way.



```
▼<category>
   <pattern>*</pattern>
   <that>MY NAME IS ALICE WHAT IS YOURS</that>
  ▼<template>
    ▼<srai>
       CALL ME
       <star/>
     </srai>
   </template>
 </category>
▼<category>
   <pattern>*</pattern>
   <that>WHO IS THE BEST ROBOT</that>
  ▼<template>
     You think
     <person/>
     is superior to ALICE?
   </template>
 </category>
▼<category>
   <pattern>*</pattern>
   <that>WHY ARE YOU SO MEAN</that>
  ▼<template>
    ▼<think>
      ▼<set name="it">
        ▼<set name="topic">
           <person/>
         </set>
       </set>
     </think>
     Are you aware that the ALICE chat robot logs and records all converstaions?
   </template>
 </category>
```

Figure 5: Example of Artificial Intelligence Markup Language (AIML) syntax [Alicebot].

Several bots still leverage the AIML language as it has proven to provide a reasonable amount of flexibility and robustness [Artificial Intelligence Foundation]. An example of the currently available options that are using AIML language, is the Pandorabots platform [Pandorabots]. One of the earliest successful bots inspired by ALICE was the SmarterChild bot. Launched in 2000, the bot was available for chatting on American Online Instant Messenger (AIM) service. SmarterChild could respond to several everyday topics. [Godara 2016]

With the launch of Facebook bot framework, a lot of new bot builders emerged that leverage the platform. Many solutions also offer more advanced artificial intelligence and machine learning functionality.

In this chapter the necessary terms and concepts have been defined and a short overview of the history of chatbots has been looked through. In the next chapter we will describe the Kentico Cloud and Kentico EMS in more detail.

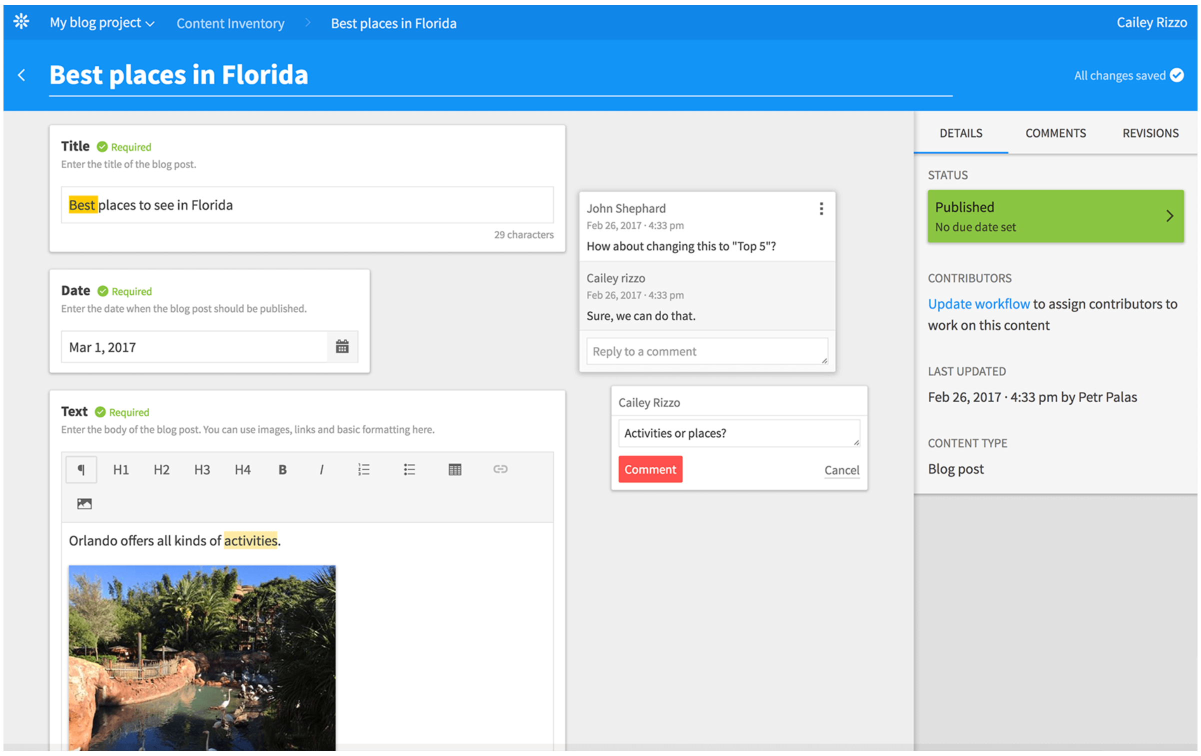
3. Kentico Cloud & Kentico EMS

For identifying the suitable bot options for content delivery from Kentico Cloud and Kentico EMS, it is necessary to describe these software to, firstly, understand what the software is, secondly, to comprehend what possibilities they offer for content publishing to different channels, and finally, to assess what integrations are possible to use with the chatbot options analyzed. Thus, the next subchapters will take a look at Kentico Cloud and Kentico EMS in more detail.

3.1. Kentico Cloud

Kentico Cloud is a cloud-first headless, platform-agnostic content management system (CMS) offered as a multi-tenant Software as a Service (SaaS). The platform allows users to create and manage structured content for multi-channel delivery. Kentico Cloud also stores assets in a Content Delivery Network (CDN). CDN consists of several geographically distributed servers where the files are stored [Wikipedia$_a$]. Storing the assets in a CDN makes the content delivery faster, as the content is delivered from the closest server. The idea of the headless CMS is to remove the front-end from the content management to defer it to the application that will display the content: headless CMS will contain only the structured content to be delivered via an API to the application handling the presentation of the content [Kentico Cloud].

Kentico Cloud allows one to create content models to specify the structure of the content items by using a drag-and-drop interface to add the necessary contents elements. The available content elements are text, rich text, number, multiple choice, date and time, asset, and modular content. It is possible to define different properties for the content elements, such as, element guidelines, make the element required, limit the length of the text input or the number of assets and so forth [Uhlar$_a$; Uhlar$_b$].



Best places in Florida

Title Required
Enter the title of the blog post.
Best places to see in Florida
29 characters

Date Required
Enter the date when the blog post should be published.
Mar 1, 2017

Text Required
Enter the body of the blog post. You can use images, links and basic formatting here.
Orlando offers all kinds of activities.

John Shephard
Feb 26, 2017 · 4:33 pm
How about changing this to "Top 5"?

Cailey rizzo
Feb 26, 2017 · 4:33 pm
Sure, we can do that.

Reply to a comment

Cailey Rizzo
Activities or places?

STATUS
Published
No due date set

CONTRIBUTORS
Update workflow to assign contributors to work on this content

LAST UPDATED
Feb 26, 2017 · 4:33 pm by Petr Palas

CONTENT TYPE
Blog post

Figure 6: View of content item in Kentico Cloud user interface [Kentico Cloud$_d$].

Once the content models are created, content items based on the models can be created where the actual content of content will be inserted as can be seen in Figure 6. During the content creation process it is possible to perform several actions: the content item can be moved to another step in a defined workflow, assigned to user(s), commented on, location in the sitemap can be defined, or a previous versions of the content item can be viewed and restored. [Kentico Cloud$_d$; Kentico Cloud$_b$]

When the content item is published it can be retrieved via the Delivery API[4]. For published items web hooks can be defined to notify external applications off updates on the items. This can be used for example to clear cache of an application to refresh content to reflect the newly updated version.

Total of three SDKs are currently available, with more coming[5], for delivering content to custom applications from Kentico Cloud [Kentico Cloud$_e$].

Next, a brief overview of Kentico EMS is offered in order to understand the difference between the two software.

[4] The Delivery API also offers a preview for unpublished content items. For more information on the Delivery API, see [Kentico Cloud$_a$].

[5] Kentico Cloud Roadmap as well as GitHub show that PHP SDK is going to be released in Q4. For more information, see [Kentico Cloud$_f$; GitHub$_b$].

3.2. Kentico EMS

Kentico EMS is an all-in-one solution offering CMS, e-commerce, and online marketing platform. It provides a wide variety of Web Content Management (WCM) features out of the box, online marketing, e-commerce, online communities, intranet, and collaboration (for a list of all features, see, [Kentico$_a$]).

Additionally, the platform is open to extension and many aspects can be easily modified to fit custom requirements. It offers open API to directly call Kentico features from a third-party application. The documentation provides a large number of examples. [Kentico$_c$; Kentico$_d$]

The built-in REST API allow reading, creating, updating, and deleting virtually any object or document in Kentico EMS. The REST API can be used for both transferring data to and from the system.

The most relevant area of Kentico EMS in the scope of this thesis is the content management. To create content, one must first define the content structure. This is done by defining page types, which consist of required fields for the content. In addition, a page template can be created or one of the built-in page templates can be used. Content items can be created using a built-in application called Pages. In the Pages application, one can input the content for the content item and place the new page in a proper hierarchy on the website. [Kentico$_b$]

In addition to content management, Kentico EMS serves the website by using Microsoft Internet Information Services (IIS) web server. Furthermore, it is possible to utilize Microsoft Azure platform to host Kentico in the cloud. [Kentico$_f$]

Vast amount of additional functionality is also offered in Kentico, however, in the scope of this thesis it is not necessary to extensively cover all of it[6]. In the next chapter, the thesis moves on to describe the available options for creating chatbots that can be utilized in delivering content.

[6] For further details on available features, see [Kentico$_e$; Kentico$_d$].

4. Available Solutions

The chatbot technology and conversational interfaces have become more and more popular with personal assistants, such as Siri, Cortana and Alexa. As mentioned before, the increasing popularity of bots can be perceived in phenomena such as the introduction of Facebook bot platform and the growth seen within a year of the platform's introduction [Facebook Newsroom 2016]. Facebook reported the rounded number of developers and bots to be 100 000 each. Other major companies are also investing a lot of time and money into bot technology [Ramos 2017]. This chapter presents different type of platforms and services available for creating bots. Some of the solutions offer a full bot builder experience while others offer some components as services that can be utilized with other solutions.

The solutions could be categorized in many ways but for the purposes of this thesis they are divided into platforms and frameworks.

4.1. Platforms

4.1.1. Dialogflow

Dialogflow is a platform owned by Google that allows you to create a natural language interface by providing actionable data based on the input given. The platform includes speech recognition, natural language understanding, machine learning as well as text-to-speech capabilities. The platform works on the basis of intents and entities recognized from the user's utterances rather than on a predefined flow pattern branching only based on the response of the user. There is a web-based user interface for defining the entities, intents, and responses for the chatbot or for other natural language interface. The agent's[7] dialog will move forward based on the identified intents that represent the intentions of the end-user. The flow of the dialog can be defined by configuring contexts, prioritizing intents, slot filling, responsibilities, and fulfillment by using web hooks, see Figure 7. [Dialogflow$_b$]

[7] The bots are referred to as agents in Dialogflow.

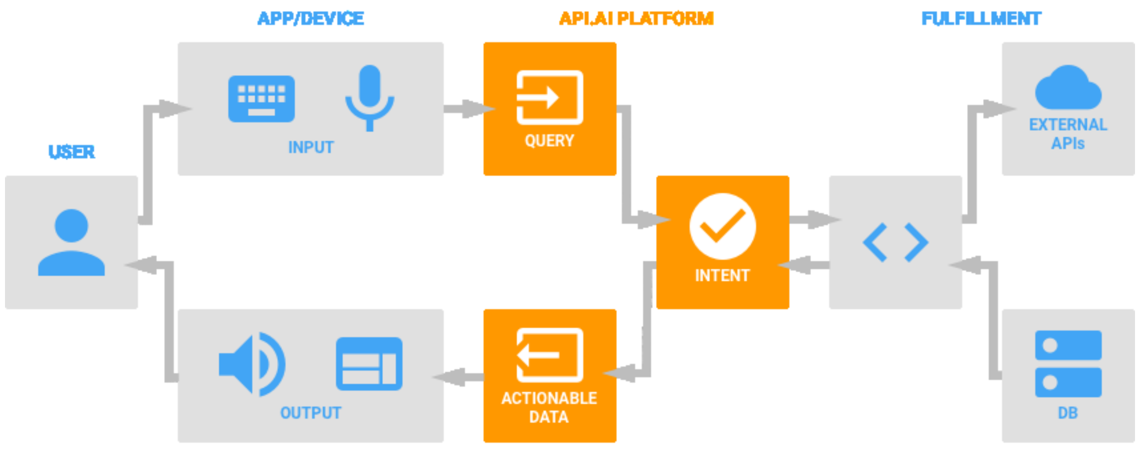


Figure 7: Overview of Dialogflow [Diagonflowa].

Dialogflow includes machine learning capabilities to further improve the detection of the intentions from the user utterances. Intents include the following sections: user says, action, response and contexts. User utterances can be written either in example mode or template mode. Example mode utterance is written in natural language and can be annotated for parameters. In the template mode, the parameters are directly referenced in the utterances and they cannot be annotated. It is recommended to rather use the example mode, because the example mode is easier to use and the machine learning can learn faster in this mode. Contexts can be used to pass information from previous conversations or external sources. For the intent to be triggered, all the contexts defined for the intent must be active. It is possible to prioritize the intents in case several intents are identified, define fallback and follow-up intents, and define text responses. Rich responses can be used in case of using one of the following one-click integrations that supports rich responses: Facebook Messenger, Kik, Slack, or Telegram. [Dialogflowi; Dialogflowe; Dialogflowf]

When defining entities, it is possible to input several synonyms so multiple words can be matched to the same entity. There are three types of entities supported by Dialogflow: system, developer, and user entities. System entities are predefined entities, which enables one to handle most common concepts, while developer entities allow one to create custom entities, and user entities are defined on the session level. Additionally, automatic expansion allows the agent to match values to entities even if they are not explicitly listed. The accuracy of matching the unidentified values gets better with adding more synonyms for the entity. The entities can be copied or moved to another agent easily through the UI. Exporting the entities in JSON or CSV format and uploading them back is also supported. [Dialogflowd]

Usage of intents and entities comes with some challenges, such as carrying over the context of the conversation. Allowing the context of the conversation to be carried over to the API permits defining the carried over context as input and/or output context for entities. Once an intent has been identified from the user utterance it can be set as the output context for future interactions. The input context is the context that needs to be set in order for an intent to be matched. If a given input context is not set, the intent will not be matched. The training section for the agent offers an option to train the agent based on the previous conversations. The analytics section offers some basic data of the usage of the agent, such as most used intents, number of sessions, and number of inquiries per session. Provided statistics for the intents include the percentage of users who exited the conversation during the intent. Additionally, and agent response time information is provided. [Dialogflow$_c$]

One-click integrations	SDKs
Google Assistant	Android
Facebook Messenger	Apple (iOS / Watch OS / Mac OS X)
Slack	Cordova
Dialogflow Web Demo	HTML
Kik	JavaScript
LINE	Node.js
Skype	.NET (WP8, WP10)
Spark	Unity
Telegram	Xamarin
Tropo	C++
Twilio	Python
Twilio Programmable Chat	Ruby
Twitter	PHP
Viber	Epson Moverio
Amazon Alexa (exporter/importer)	Botkit
Microsoft Cortana (exporter)	Java

Table 1: Dialogflow integrations and SDKs [Dialogflow$_e$; Dialogflow$_k$].

A s seen in Table 1, the platform offers a wide range of integrations and SDKs. Additionally, Dialogflow provides a variety of prebuilt agents, which can be customized to fit your needs.

Dialogflow currently supports 15 languages[8], which can used for the bots [Dialogflow$_h$; Dialogflow$_j$].

[8] Brazilian Portuguese, Chinese (Cantonese, Simplified & Traditional), English, Dutch, French, German, Italian, Japanese, Korean, Portuguese, Russian, Spanish and Ukrainian

While it is not possible to directly clone an agent, the agent can be exported and after creating a new agent, the previous agent's export can be restored to the new agent in order to transfer the entities and intents.

4.1.2. Chatfuel

Chatfuel is a bot platform utilizing the Facebook Messenger platform and Telegram. However, Chatfuel does support integration with services such as IFTTT (If This Then That) and Zapier in order to allow, for example, to adding a new card from Instagram updates. In addition to offering the integrations, there is a JSON API available for creating custom integrations with other applications, which makes the platform more flexible. The API allows to make GET and POST requests to fetch the data and extract information from responses. Through the API it is possible to generate dynamic content, get and set user attributes, redirect users to another block in the bot and create postbacks.[9] [Chatfuel$_a$]

The bot comprises of a set of blocks that allow the user to define different actions based on the users input. The bot will move to another block depending on the answer that is received from the user. Blocks are built by using plugins that can, for example, move to another block, show content from RSS feed, perform a Google search or get input from the user. Chatfuel offers the following types of responses: text, image, gallery, list, quick reply, and typing[10]. The flow of the conversation is defined by the user replies and the branches based on them, thus, the entire conversation flow is predefined. The blocks are connected to each other by using the plugins inside the block, but there is no overview showing all the connections between the blocks. Nevertheless, in order to organize the modules and provide structure for the bot, it is possible to group the modules into named groups. [Chatfuel$_e$]

The user input can be text, images or files supported by Facebook Messenger such as .pdf or .docx files. The input can be parsed and validated by the User Input Plugin, which saves the input to a given variable for the future use in the conversation. [Chatfuel$_f$] Chatfuel does not offer any NLP capabilities on its own and as such, the interaction with the bots can easily become frustrating or the user might lose interest as the bot follows a strict pattern to advance the

[9] For more details on the JSON API, see [Friedrich].

[10] This response type shows the message "…", which indicates that the other party of the conversation (i.e. the bot) is typing.

conversation. However, as Chatfuel uses Facebook Messenger platform, it can benefit from the built-in NLP functionalities. This functionality is, nevertheless, quite limited in its nature and only supports English language by default. Moreover, the detected entities are limited to greetings, thanks, goodbyes, dates, times, locations, amounts of money, phone numbers and emails. Facebook Messenger platform offers a possibility to customize the NLP through wit.ai by adding custom entities in English as well as other languages. [Facebook for Developers]

4.1.3. Motion.ai

Motion.ai is a platform that offers a visual builder for creating flow-based bots by connecting modules. Additionally, Motion.ai offers a NLP engine, which is currently available as beta, in order to understand natural language and introduce the idea of intents and entities to the platform. Combining the flow-based dialog and intents allows for more flexibility for the user as they can interact more naturally with the bot since it remembers the context of the previous messages. However, the documentation on this feature is quite limited in the beta stage. There is a video series that details how to enable the NLP engine, create intents, tag for entities and train the NLP engine. The intents should be trained by providing example user utterances, which allow the NLP engine to recognize the intents and entities with more confidence. [Motion.ai$_f$; Motion.ai$_e$]

Motion.ai does not provide a way to define input and output contexts in the same way as Dialogflow does. Each module can be connected to one or more modules and each of the connections acts as an if-condition. This allows for branching based on the response of the user, extracted data, or custom variables. Finally, a default fallback option can be provided, if none of the defined options match. [Motion.ai$_a$]

Motion.ai has a list of standard modules for common tasks, which are available out of the box. Additionally, Node.js module allow using Node.js code in bots to, for example, make calls to an external service or run other custom code. The Node.js module is expected to return a JSON object with a certain structure in order for the Motion.ai platform to properly process the output of the custom module. The module allows including several popular libraries in order to extend the code. In case the NLP engine is enabled, the NLP enriched data is available for use in the Node.js module in addition to the standard data available in the response [GitHub$_d$]. List of available modules along with their descriptions can be seen in Table 2.

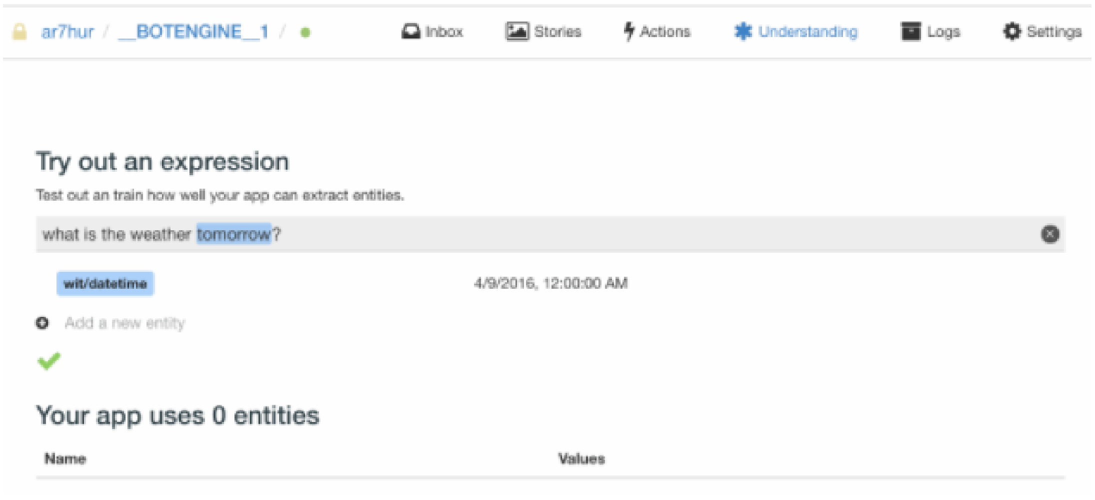
Module	**Purpose**
Bot Statement	The simplest of modules, bot statement is best for situations when you have no expectation of what the user's response might be. Think of it as a free-form text field.
Multiple Choice	Intended for situations where a number of defined options or answers exist to a given question.
Sentiment (Yes/No)	Formatting and extraction of data for Yes / No questions.
Email Collection	Extracts an email address from an abstract response
URL Collection	Extracts a URL from an abstract response
Number Collection	Converts both numeric and phonetical numbers into integers.
Phone Number Collection	Extracts and validates telephone numbers based on selected country codes.
Date/Time Collection	Extracts and validates an ISO string date stamp
Address Collection	Extracts and validates a geolocation address, if detected
Duration Parsing	Extracts representations of time and parses them into total seconds
Get Name	Extracts the user's name from user input

Table 2: Available modules in Motion.ai [Motion.aic].

In regard to channels, Motion.ai offers the following options: web chat, SMS, Facebook Messenger, Slack, and email. In case of any other medium, a REST API is also offered for broadcasting the messages from the bot [Motion.aib].

4.1.4. Wit.ai

Wit.ai platform is owned by Facebook, and it is a similar service as above-described Dialogflow, since it offers a service that provides actionable data based on user input. Instead of providing a visual builder to design the conversation flow, wit.ai offers an extensive API for parsing the input and providing NLP parsed information for custom applications. As the platform is owned by Facebook, there is an existing integration in Facebook Messenger platform for using use wit.ai. [Facebook for Developers]. For integrating with other applications, one must handle sending requests and parsing the information returned by wit.ai in their custom application. The conversation is managed by building stories, which consist of entities and intents. There is a good amount of built-in entities, which are optimized for use in the applications. Hence, it is recommended to use them whenever possible. In case there are no suitable built-in entities it is also possible to create custom entities to extend the possibilities. The built-in entities can be easily used by first giving an example user utterance, marking the appropriate part of text as the entity and selecting the entity type, as can be seen in Figure 8 [wit.aia].



Try out an expression

Test out an train how well your app can extract entities.

what is the weather tomorrow?

wit/datetime 4/9/2016, 12:00:00 AM

Your app uses 0 entities

Figure 8: Tagging utterances for entities in wit.ai [wit.aid].

Custom entities can also be defined in the same way and there are different options to be configured for the custom entity, depending on the lookup strategy selected (trait value, keywords, or synonyms). The possible options for lookup strategy along with use cases and examples can be seen in Table 3. The trait value and the keyword represent the value API will return upon extracting the entity. Synonyms can be defined for the keywords in order to allow for several words to be matched to a single keyword. [wit.aid]

Wit.ai API is quite extensive and offers clear examples for each available API command along with the sample response that would be returned. The app can be trained via the API by providing sample utterances along with the entities, which should be recognized once the training finishes. The platform provides context objects to allow for the application to remember the state of the current conversation. The context can only be updated on the client side application. Wit.ai can use this information to decide between different responses as is defined in the stories, which allows more context aware responses. [wit.aid]

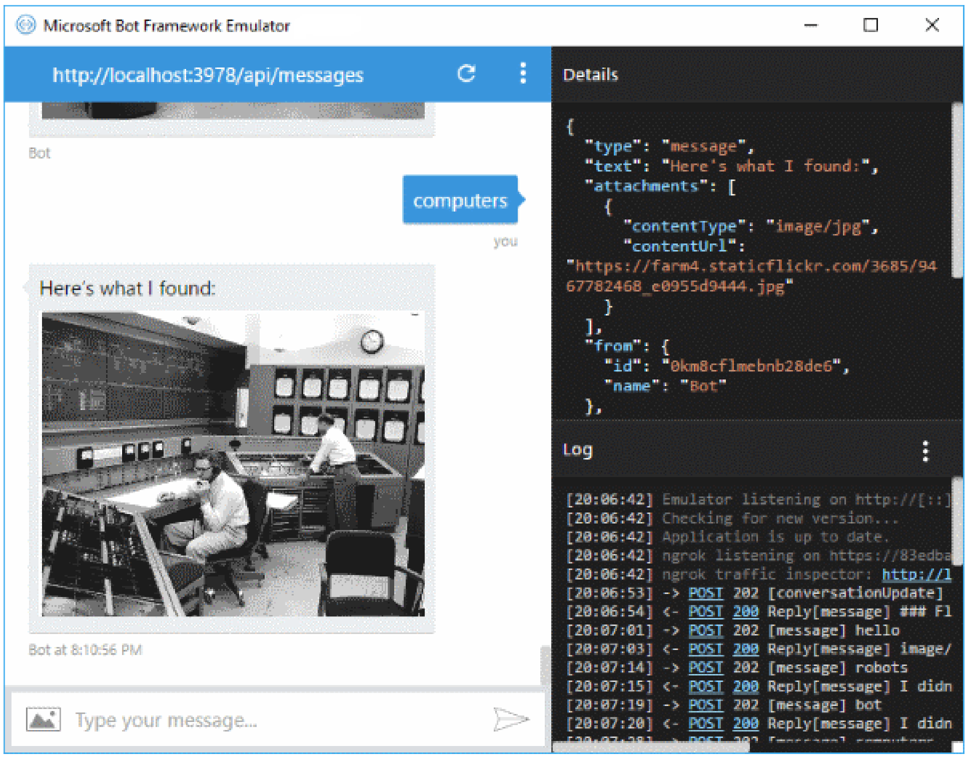
Lookup Strategy	Use Case	Examples
Trait	When the entity value is not inferred from a keyword or specific phrase in the sentence. There is no obvious association between certain words in the sentence and the value of the entity, but rather you need the sentence as a whole to determine the value.	Intent, Sentiment, politeness
Free Text	When you need to extract a substring of the message, and this substring does not belong to a predefined list of possible values.	Message Body, Contact Name
Keywords	When the entity value belongs to a predefined list, and you just need substring matching to look it up in the sentence.	Country, Burger, Room

Table 3: Types of entities for wit.ai [wit.ai$_d$].

4.2. Frameworks

4.2.1. Microsoft Bot Framework, Microsoft Language Understanding Intelligent Service (LUIS), and Cognitive Services

Microsoft's Bot Framework enables one to create, test, connect, and deploy bots. It offers an option to get started quickly with the bot development by using Azure Bot Service to speed up the development by offering a web host and ready-made bot templates that can be used as a base to develop a custom bot. [Brandl and Standeref 2017] In addition to this, it is possible to use the Bot Builder SDK available for .NET and Node.js to build a bot from the ground up. The Bot Builder SDK provides an emulator that allows for easy testing and debugging of the bot solution locally, as seen in Figure 9. [Mak et al. 2017; Standefer et al. 2017a; Standefer et al. 2017b] In addition, Bot Framework REST API is offered to make development in other programming languages possible.



```
{
  "type": "message",
  "text": "Here's what I found:",
  "attachments": [
    {
      "contentType": "image/jpg",
      "contentUrl":
"https://farm4.staticflickr.com/3685/94
67782468_e0955d9444.jpg"
    }
  ],
  "from": {
    "id": "0km8cflmebnb28de6",
    "name": "Bot"
  },
```

```
[20:06:42] Emulator listening on http://[::]
[20:06:42] Checking for new version...
[20:06:42] Application is up to date.
[20:06:42] ngrok listening on https://83edba
[20:06:42] ngrok traffic inspector: http://1
[20:06:53] -> POST 202 [conversationUpdate]
[20:06:54] <- POST 200 Reply[message] ### F1
[20:07:01] -> POST 202 [message] hello
[20:07:03] <- POST 200 Reply[message] image/
[20:07:14] -> POST 202 [message] robots
[20:07:15] <- POST 200 Reply[message] I didn
[20:07:19] -> POST 202 [message] bot
[20:07:20] <- POST 200 Reply[message] I didn
```

Figure 9: Microsoft Bot Framework Emulator running locally for debugging [Mak et al. 2017].

Microsoft Bot Framework offers a myriad of samples[11] showcasing the features provided by the framework: these range from single feature samples to a more complete reference implementation. The Bot Framework offers advanced tools for handling the dialog flow, for example, sending and receiving text responses and attachments, remembering the context of the conversation, and integrating other services such as search engines and natural language understanding. The platform offers a wide documentation and API reference as well as structured sections for different phases of the development ranging from the design of the bot to deploying and connecting the bot to different channels. If one chooses to use the Azure Bot Service, one can also use one of the ready provided templates to quickly get started on the bot development. A list of provided templates can be seen in Table 4.

[11] For a comprehensive list of samples, see [Standefer et al. 2017a; Standefer et al. 2017b].

Template	**Description**
Basic	Creates a bot that uses dialogs to respond to user input.
Form	Creates a bot that collects input from a user via a guided conversation that is created using FormFlow (in C#) or waterfalls (in Node.js).
Language understanding	Creates a bot that uses natural language models (LUIS) to understand user intent.
Proactive	Creates a bot that uses Azure Functions to alert users of events.
Question and Answer	Creates a bot that uses a knowledge base to answer the user's questions.

Table 4: Built-in templates offered for Azure Bot Service [Brandl and Standeref 2017].

In regard of the available channels for the bot, one can choose from a wide variety of options including Bing, Cortana, Skype and Skype for Business, Microsoft Teams, Web Chat, Email, GroupMe, Facebook Messenger, Kik, Slack, Telegram, and Twilio. In addition to the mentioned channels, one can use Direct Line to connect the bot to their own application. The available options for each bot depend on the features used by the bot as not all platforms support all the response types offered by Microsoft Bot Framework. A Channel Inspector is offered to make it easier for developers to see which response types are offered and how the features look on each of the available channels. Microsoft Language Understanding Intelligent Service (LUIS) can provide bots[12] with additional data to understand the user input. The service provides as a response the intents and entities extracted from the user input. In addition to the intent and entity recognition offered by LUIS, one can utilize the more complex services offered by Cognitive Services to extract meaningful data from user input, as can be seen in Table 5. By using Azure Bot Service, one can easily start utilizing some of the mentioned Cognitive Services. Nevertheless, it is also possible to integrate them into a bot that is created using the Bot Builder SDK. A few examples on integrating these services are offered in the sample implementations.

[12] Naturally Microsoft Language Understanding Intelligent Service (LUIS) can be used for other types of applications as well, but in the scope of the thesis it is not relevant.

Computer Vision	Linguistic Analysis API
Content moderator	Recommendations API
Video API	Knowledge Exploration Service
Face API	Entity Linking Intelligence Service
Emotion API	Academic Knowledge API
Video Indexer	QnA Maker API
Custom Vision Service	Custom Decision Service
Translator Speech API	Bing Autosuggest API
Bing Speech API	Bing News Search API
Speaker Recognition API	Bing Web Search API
Custom Speech Service	Bing Entity Search API
Bing Spell Check API	Bing Image Search API
Web Language Model API	Bing Video Search API
Text Analytics API	Bing Custom Search
Translator Text API	

Table 5: Microsoft Azure Cognitive Services offered [Microsoft Azure].

4.2.2. IBM Watson

IBM Watson framework is similar to Microsoft Bot Framework, which offers a variety of different services that can be used to build versatile conversational agents. A list of the services offered can be seen in Table 6. The platform offers an easy option for deploying a pre-trained virtual agent for certain domains, which allows one to quickly launch a bot for customer service purposes. Building a conversational agent is also possible by utilizing the SDKs available for Node.js, Python, Swift, Java, Unity, and .NET. If a SDK is not available for a programming language, there is a REST API can be used. [GitHub$_f$]

Conversation	Quickly build and deploy chatbots and virtual agents across a variety of channels, including mobile devices, messaging platforms, and even robots.
Virtual Agent	Build a chatbot for customer service - no machine learning experience required.
Visual Recognition	Tag, classify and search visual content using machine learning.
Discovery	Unlock hidden value in data to find answers, monitor trends and surface patterns with the world's most advanced cloud-native insight engine.
Natural Language Understanding	Analyze text to extract meta-data from content such as concepts, entities, keywords and more.
Discovery News	Infuse dynamic news content into every app you build.
Knowledge Studio	Teach Watson to discover meaningful insights in unstructured text without writing any code.
Document Conversion	Document Conversion capabilities have been migrated to Watson Discovery. Take advantage of improvements to PDF conversion using Watson Discovery.
Speech to Text	Convert audio and voice into written text for quick understanding of content.
Text to Speech	Convert written text into natural sounding audio in a variety of languages and voices.
Personality Insights	Predict personality characteristics, needs and values through written text.
Tone Analyzer	Understand emotions, social tendencies and perceived writing style.
Language Translator	Translate text from one language to another.
Natural Language Classifier	Interpret and classify natural language with confidence.
Retrieve and Rank	Retrieve and Rank has evolved into Watson Discovery and is deprecated as a stand-alone service.

Table 6: Services offered by IBM Watson framework [IBM Watson].

API documentation offers an overview of the available commands along with examples on how to make the request in different programming languages and on sample responses that would be received from the API. In regard to chatbots, the platform works based on intents and entities. The dialog flow can be defined by using a visual tool to insert nodes into a tree structure. Each node works as a conditional node: when given input X and the condition Y is fulfilled, perform action Z. The action performed can be to return a response or to trigger a programmatic action, as can be seen in Figure 10.

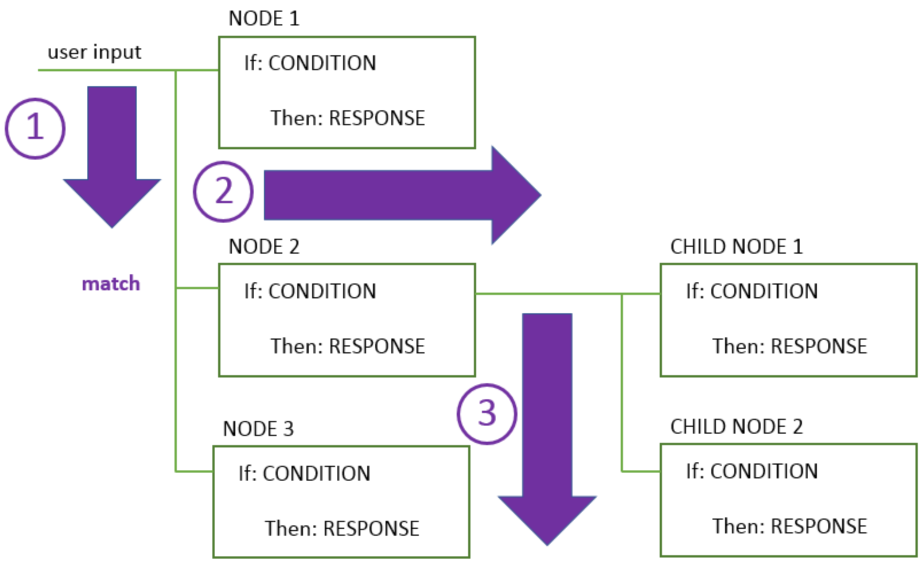


Figure 10: IBM Watson dialog flow illustration [IBM Bluemix 2017b].

When it comes to deployment, there are differences in IBM Watson in comparison to previous options. Test deployment can be done to Slack by using the test deployment tool offered by the platform. However, to deploy the IBM Watson bot to social media channels, such as Slack, Facebook Messenger and Twilio, one must use Botkit middleware.[13]

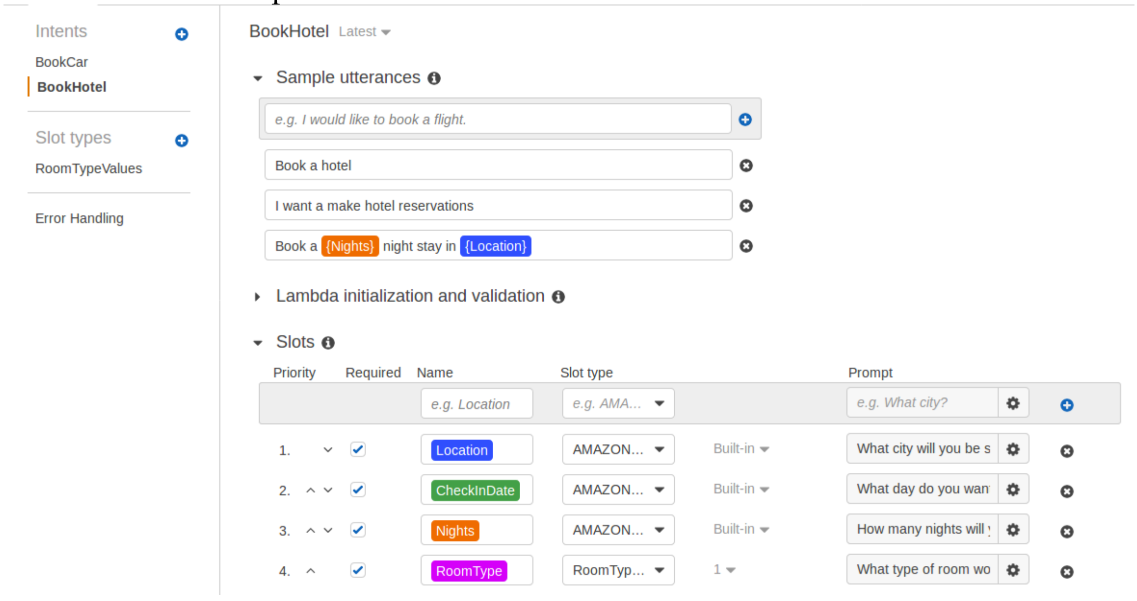
4.2.3. Amazon Lex

Amazon Lex is an Amazon Web Services (AWS) service that offers easy access to the technology used in Alexa personal assistant. Amazon Lex allows one to create a conversational interface for any application, which uses voice or text input. The service includes, among other things, natural language understanding to enable natural conversions and machine learning to enable the ability to improve through interactions. Amazon Lex also benefits from the other services offered by AWS. They enable for more flexibility and complex implementations, which can also scale when needed.

[13] More information on Botkit middleware, see [IBM Bluemix 2017a; GitHube].

Amazon Lex works by identifying the intents of the user from the input and filling the required slots in case there are any required slots defined. Slots are similar to entities, and they represent values that are necessary for fulfilling the intent of the user. The necessary slots are defined in the intent along with a prompt text, which Lex will use to query said information. Amazon provides a very large number of possible slot types for gathering the necessary information in order to fulfill an intent.[14] In addition to the built-in slot types, it is possible to define a confirmation prompt for each intent and a AWS Lambda functions for validating the user input and fulfillment. Alternatively, one can configure the bot to return the slot information to the application for custom processing. [Amazon Developer]

Amazon Lex does not provide an interface for defining the dialog flow, as the dialog will advance based on the intents identified from the user input. As the dialog progresses the user is prompted to fill the required slots. Several slots can be filled from single user input, which, as discussed earlier, allows for a more flexible user experience.



e.g. I would like to book a flight.

Book a hotel

I want a make hotel reservations

Book a {Nights} night stay in {Location}

Priority	Required	Name	Slot type		Prompt
		e.g. Location	e.g. AMA...		e.g. What city?
1.	✓	Location	AMAZON...	Built-in	What city will you be s
2.	✓	CheckInDate	AMAZON...	Built-in	What day do you wan
3.	✓	Nights	AMAZON...	Built-in	How many nights will :
4.	✓	RoomType	RoomTyp...	1	What type of room wo

Figure 11: Example of intent configuration in Amazon Lex [Amazon Web Services$_a$].

Amazon Lex offers mobile SDKs for iOS, Android, Xamarin, and Unity. Integrations are available for Facebook Messenger (text input only), Slack, and Twilio. Additionally, bots can be exported as Alexa Skills in order to use them with Amazon Alexa[15].

[14] For a full list of available slot types, see, Amazon Lex documentation [Amazon Developer].

[15] It requires some extra steps for the skill to be properly defined. For more details, see, [Amazon Web Services].

5. Delivering Content

This chapter determines how the presented services fit the purpose of delivering content from Kentico EMS and Kentico Cloud. To do so, it is necessary to establish the requirements from the point of view of the source of the content. In order to achieve this goal, criteria based on which it is possible to assess the suitability of each option, has been chosen. Additionally, the chapter takes a look at Kentico EMS and Kentico Cloud to consider if there is functionality missing that would prevent or make it harder to integrate the products with the chatbot options.

The goal of the thesis is not to select a single option but rather initially evaluate the options to identify what kind of features are available and what they might require from the Kentico products. The focus is on what are the necessary features or properties a chatbot service should have to be practical for content delivery.[16] The following criteria is used to evaluate the options:

1. Is the product in active development?
2. Does the product have an active community?
3. Is the documentation of the products comprehensive?
4. How easy is it to integrate with channels?

The first two criteria have been selected to ensure that the option is not showing signs of being abandoned either by the developer (item 1) or by the community using it (item 2). Furthermore, an active community can give an idea of the support that is received from the developing part: if the community forums are actively answered by the company representatives, it shows a level of commitment to the product. The third and the fourth criteria are used to determine how easy the development would be and to evaluate the ease of integration with other third-party products. Moreover, a comprehensive documentation that offers a good number of examples can help reduce the initial learning curve when starting to use the product, thus, likely leading to faster development.

[16] For evaluating the quality of chatbots and conversational agents, see [Radziwill and Benton 2017].

5.1. Platforms

5.1.1. Dialogflow

Dialogflow was originally known as Speaktoit and, until recently, as API.ai[17]. The company was acquired by Google in September 2016. The product is being actively developed by the company, and new features and sample bots are being added to the service. The product has a GitHub repository for sample implementation, however, it does not seem to be very active. On the other hand, the community forum appears to be relatively active with several topics posted each day.

Dialogflow's documentation is well organized and includes a good number of examples and how–to guides. For most of the questions the answers can be found in the documentation. The user interface of Dialogflow also includes a good number of helpful tooltips and descriptive texts, which gives the user an easy access to help without the need of navigating to the documentation. The built-in integrations that are offered are simple to enable. For fulfillment, one can define webhooks to fetch content from an external application, such as Kentico Cloud or Kentico EMS. Recently, Dialogflow added the possibility of defining the fulfillment logic in the web user interface by utilizing Google's Cloud functions for Firebase. Additionally, webhooks can be defined to communicate with third-party applications.

We can determine that Dialogflow is an active, stable product backed by a large company, with comprehensive documentation, and offers good examples. The platform has a good amount of functionality, which supports a wide variety of different kind of bots and channels to broadcast content to. However, no REST API is provided with the platform and as such, for example, pushing newly created content without user interaction from Kentico EMS or Kentico Cloud is not possible. Using Dialogflow for a chatbot, that pulls content from Kentico EMS or Kentico Cloud is certainly viable. The number of easily enabled channels makes Dialogflow a good option especially for situations where one want to make existing content available through several channels.

5.1.2. Chatfuel

Chatfuel is a platform that is targeted mainly for the Facebook Messenger platform; however, it started in 2015 on Telegram messenger. The option for creat-

[17] October 2017 API.ai was renamed to Dialogflow, and new features were added to the platform. For more details, see [Dialogflow$_g$].

ing a bot for Telegram still exists, but the main focus is on the Facebook Messenger platform. While the platform hasn't been around for a long time, according to Mindbowser Chatbot Survey in 2017 [Mindbowser 2017], it is the fourth most popular chatbot platform as ranked by respondents when asked for the best platform to build chatbots with.

The company is actively developing the product and posting new blog posts several times a month, which detail new features or give hints on using the existing features of the platform. Additionally, the community forums for Chatfuel seem to be active with community members contributing to the platform, for example, by sharing custom tools they have created to ease the development on the platform[18].

Documentation for the platform is quite limited considering there were only 44 articles at the time of the writing[19]. While the company is fairly active in posting guides on current and new features on its blog, the overall amount of documentation is unfortunately very limited. The documentation is missing a clear navigation page, which would show the hierarchy of articles. Search functionality is available for searching the documentation, but it does not provide a quick overview of the documentation like a hierarchical structure would. Nevertheless, the company has recently been active in other ways, such as, publishing an electronic book in collaboration with HubSpot, which details how one can build their first bot with Chatfuel platform.[20]

Chatfuel offers a way to broadcast messages to existing users who have previously interacted with the bot. The user interface allows for defining follow-up messages and scheduling messages to be broadcasted at a later time. Additionally, throught IFTTT and Zapier integrations it is possible to set up autoposting from external sources. This feature makes Chatfuel able to deliver content from Kentico EMS and Kentico Cloud as webhooks could be set up to autopost new content that has been added to the system. The platform also offers a JSON REST API that enables one to communicate with third-party custom applications which allowing chatbot responses to be constructed by external application.

Chatfuel does offer a good amount of functionality, even though it is constrained by the Facebook Messenger platform. If one is looking for a simple way to broadcast content through Facebook Messenger platform, Chatfuel is an

[18] For an example of community user provided tool available on the forums, see [Chatfuel$_d$].
[19] Articles available in documentation, see [Chatfuel$_c$].
[20] The book can be downloaded from Chatfuel's blog, see [Chatfuel$_b$].

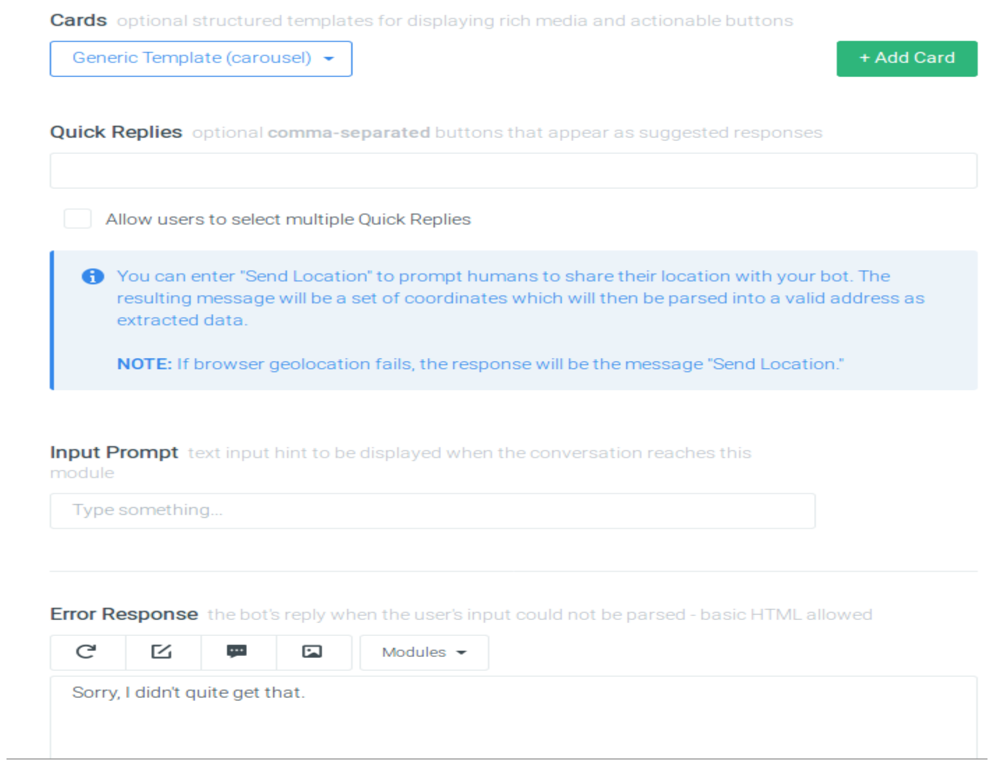
option that would suit this scenario. Nevertheless, for more advanced scenarios one might utilize the customization option available through wit.ai, which is available through Facebook Messenger platform.

5.1.3. Motion.ai

Motion.ai platform was launched in 2015, and it states as its goal to offer a single easy-to-use service for creating chatbots that can serve content to multiple channels. [Motion.ai$_c$] Motion.ai was recently acquired by HubSpot, and for now the offered service remains as it is. However, there is no promise of how long the service will be provided in its current state. This poses a possible issue since the service is meant as the single application to fulfill all the needs instead of integrating with other third-party systems. In the event that the service would be discontinued and, for example, if the service would only be offered as a part of the HubSpot CMS software, it would present risk of a vendor lock-in.

Regarding community, Motion.ai does not currently offer forums, but a slack channel is mentioned on the company's website. Documentation for the platform is nicely organized in categories, but it offers a quite limited number of topics. Nevertheless, this is partially compensated by the design of the user interface. As seen in Figure 12, the service is filled with hints and descriptive texts that allow the user to find out most of the necessary information without the need to navigate to the documentation in the first place. Additionally, the company has prepared an example available on GitHub of using the Node.js modules in a bot as well as a series of YouTube videos[21] detailing how to utilize the NLP engine and set up intents and entities for it. Motion.ai, similarly to Chatfuel, does not have dedicated community forums but only a Slack channel.

[21] Video series is available on Motion.ai's YouTube channel, see [Motion.ai$_f$].


Cards optional structured templates for displaying rich media and actionable buttons

Generic Template (carousel) ▾ | + Add Card

Quick Replies optional **comma-separated** buttons that appear as suggested responses

☐ Allow users to select multiple Quick Replies

You can enter "Send Location" to prompt humans to share their location with your bot. The resulting message will be a set of coordinates which will then be parsed into a valid address as extracted data.

NOTE: If browser geolocation fails, the response will be the message "Send Location."

Input Prompt text input hint to be displayed when the conversation reaches this module

Type something...

Error Response the bot's reply when the user's input could not be parsed - basic HTML allowed

Modules ▾

Sorry, I didn't quite get that.

Figure 12: Hint and descriptive texts for Geo-Address module in Motion.ai [Motion.ai$_c$].

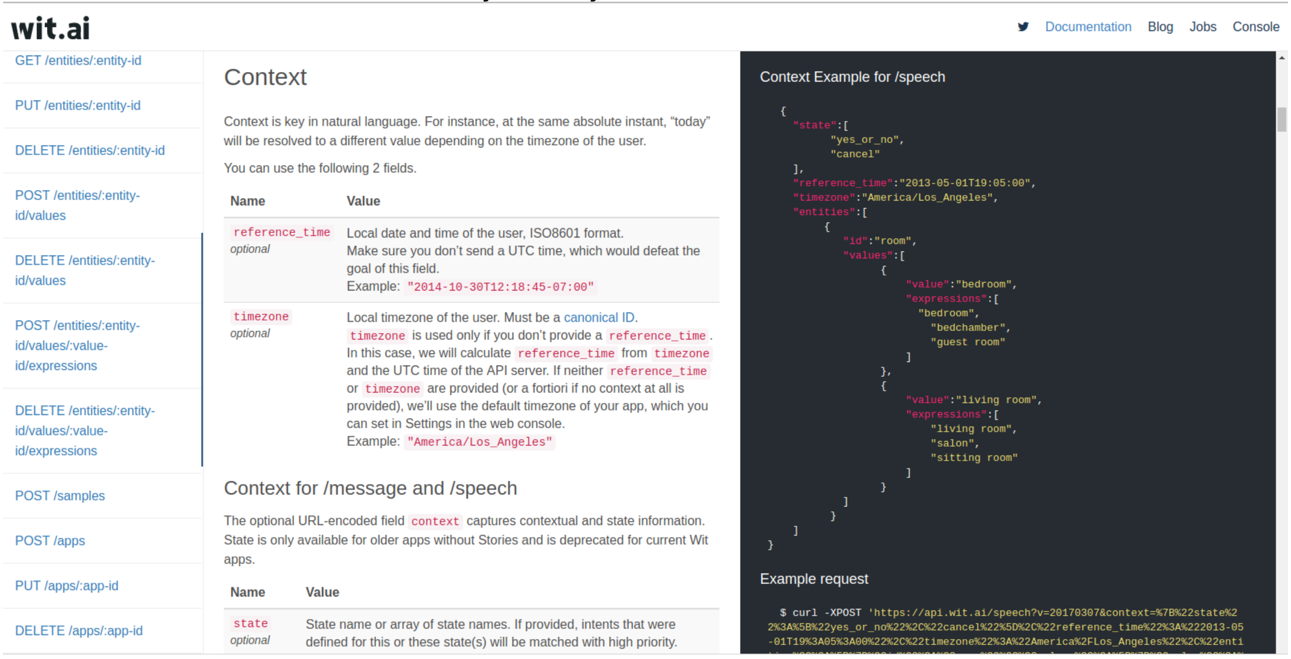
In regard to connecting the bot to social media and other channels, the service only offers a limited amount of options out of the box, and choice of the channel has to be made when starting the development of the bot. It is not possible to connect the same bot to several different channels. The offered RESTful API, however, does allow one to broadcast messages to officially unsupported channels.

Based on what has been presented in this subchapter, it is clear that the functionality offered by Motion.ai is suitable to be used for content delivery from Kentico Cloud and Kentico EMS. Through Node.js modules it is possible to make calls to Delivery API of Kentico Cloud or to the REST API of Kentico EMS to fetch the content.

5.1.4. wit.ai

The platform wit.ai was acquired by Facebook in early 2015 and has since been developed actively. In July 2017, wit.ai announced that it is going to shut down the bot builder and instead focus exclusively on providing a NLP service. The reason behind this decision was the fact that the tools offered for building bots have advanced since the bot engine was originally brought to market. At the same time as the company announced shutting down the service, they also revealed the built-in NLP service for the Facebook Messenger platform. [wit.aic]

The platform, while not offering a full bot builder, does provide a good NLP engine, which can be used in practically any application through their API. The development is going on actively, and this can be seen, for example, in the wit.ai's GitHub repository that has 26 repositories listed. The documentation for the platform is comprehensive and offers a good amount of resources for developers. It includes a getting started guide, example solutions to problems as well as an exhaustive HTTP API Reference going through all the available endpoints. Examples of the requests and sample responses returned are also shown in the reference, as can be seen in Figure 13. SDK is offered for Node.js, and libraries are available for Ruby and Python.


wit.ai
Documentation
Blog
Jobs
Console
GET /entities/:entity-id
PUT /entities/:entity-id
DELETE /entities/:entity-id
POST /entities/:entity-id/values
DELETE /entities/:entity-id/values
POST /entities/:entity-id/values/:value-id/expressions
DELETE /entities/:entity-id/values/:value-id/expressions
POST /samples
POST /apps
PUT /apps/:app-id
DELETE /apps/:app-id

Context

Context is key in natural language. For instance, at the same absolute instant, "today" will be resolved to a different value depending on the timezone of the user.

You can use the following 2 fields.

Name	Value
`reference_time` *optional*	Local date and time of the user, ISO8601 format. Make sure you don't send a UTC time, which would defeat the goal of this field. Example: `"2014-10-30T12:18:45-07:00"`
`timezone` *optional*	Local timezone of the user. Must be a canonical ID. `timezone` is used only if you don't provide a `reference_time`. In this case, we will calculate `reference_time` from `timezone` and the UTC time of the API server. If neither `reference_time` or `timezone` are provided (or a fortiori if no context at all is provided), we'll use the default timezone of your app, which you can set in Settings in the web console. Example: `"America/Los_Angeles"`

Context for /message and /speech

The optional URL-encoded field `context` captures contextual and state information. State is only available for older apps without Stories and is deprecated for current Wit apps.

Name	Value
`state` *optional*	State name or array of state names. If provided, intents that were defined for this or these state(s) will be matched with high priority.

Context Example for /speech

```
{
  "state":[
        "yes_or_no",
        "cancel"
  ],
  "reference_time":"2013-05-01T19:05:00",
  "timezone":"America/Los_Angeles",
  "entities":[
       {
         "id":"room",
         "values":[
              {
                 "value":"bedroom",
                 "expressions":[
                   "bedroom",
                     "bedchamber",
                     "guest room"
                 ]
              },
              {
                 "value":"living room",
                 "expressions":[
                     "living room",
                     "salon",
                     "sitting room"
                 ]
              }
         ]
       }
  ]
}
```

Example request

```
$ curl -XPOST 'https://api.wit.ai/speech?v=20170307&context=%7B%22state%2
2%3A%5B%22yes_or_no%22%2C%22cancel%22%5D%2C%22reference_time%22%3A%222013-05
-01T19%3A05%3A00%22%2C%22timezone%22%3A%22America%2FLos_Angeles%22%2C%22enti
```

Figure 13: Example from wit.ai HTTP API Reference [wit.aib].

According to wit.ai blog, the usage of the platform has grown from 20,000 developers to more than 100,000. The popularity of the platform can be seen in the Mindbowser [2017] chatbot survey with more than 45 percent of respondents favouring the platform.

As the platform is currently offering only NLP services, direct integration with social media is not available, with the exception of Facebook Bot platform, which uses wit.ai to offer the built-in NLP functionality. Integrating with other channels has to be handled within a custom application. The platform offers a solid NLP service that comes with a well-documented API for interactions as well as a stable company backing the service. As previously mentioned, wit.ai allows one to extract intents and entities from the user input for a more in-depth understanding of the user's intent. The NLP engine provides the developers with a good amount of supporting data to help them create a context-aware conversational interfaces for the application of their choice.

When considering the use of wit.ai from the point of view of the thesis' goal, as itself the service does not offer the kind of functionality that would be useful for content delivery. However, the service can certainly be used in conjunction with other services that provide other necessary services for interpreting the NLP data, constructing the responses based on the data and broadcasting them the channel of choice. For example, platforms that are targeting the Facebook Messenger platform, utilizing the Facebook Bot platform, can benefit from the additional flexibility that the additional NLP data offers. This allows for more natural experience when interacting with the bots.

In regard to content delivery from Kentico Cloud or Kentico EMS there is nothing that would prevent it. The content would be fetched by the custom application that is utilizing wit.ai to provide the NLP services. All parties offer REST API for communicating with third-party services, and thus it can be concluded that no restrictions apply from wit.ai's or Kentico's side. The logic for content delivery can be implemented in a custom application utilizing the REST APIs available.

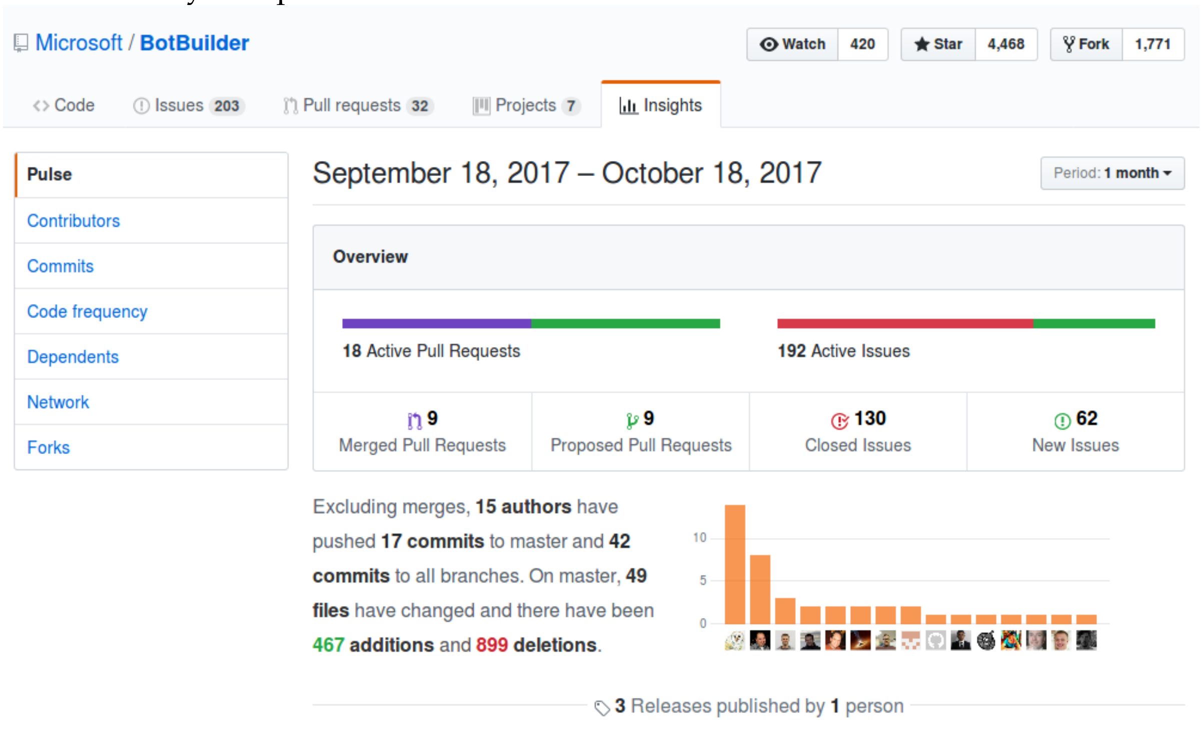
5.2. Frameworks

5.2.1. Microsoft Bot Framework, Microsoft Language Understanding Intelligent Service (LUIS), and Cognitive Services

Microsoft Bot Framework offers a very comprehensive set of features, and this can be seen in the surveys as well: for example, in Mindbowser's [2017] chatbot survey 41 percent of the respondents preferred the framework over other options. The framework has been available for the public from the end of March 2016 [Microsoft 2016]. Microsoft has been actively developing the service and adding supporting services that allow for more complex functionalities for improving the bots. For example, the Bot builder repository in GitHub shows a total of 42 commits to all branches in the past month, as can be seen in Figure 14.

The documentation for the framework is very comprehensive covering the SDKs (C# and Node.js), Azure Bot Services, and REST (Direct Line API). The documentation includes also introductions to basic concepts of bots as well as sections on designing, testing, debugging, and deploying bots. The Microsoft

Bot Framework enables one to easily connect to a large number of channels within the web user interface with a flip of a switch or by following simple configuration steps [Brandl et al. 2017]. The Microsoft Bot Framework, Microsoft Language Understanding Intelligent Service (LUIS), and the Cognitive Services are analyzed together as they are tightly connected. However, it is worth noting that the LUIS and the Cognitive Services can also be used without the Bot Framework as well. As mentioned earlier, the LUIS service is a part of the Cognitive Services, which is a collection of services providing different kind of data from a variety of input.



September 18, 2017 – October 18, 2017

18 Active Pull Requests

192 Active Issues

9 Merged Pull Requests

9 Proposed Pull Requests

130 Closed Issues

62 New Issues

Excluding merges, **15 authors** have pushed **17 commits** to master and **42 commits** to all branches. On master, **49 files** have changed and there have been **467 additions** and **899 deletions**.

3 Releases published by **1** person

Figure 14: Commits for BotBuilder repository for 18th September to 18th October 2017[GitHub$_c$].

Overall, Microsoft offers a complete set of tools for all kinds of bots from simple ones to advanced ones. However, utilizing the Bot Framework and the Cognitive Services would require more effort than many of the previous platforms that were analyzed.

Nevertheless, in case one needs a bot that is armed with powerful understanding capabilities for a variety of different types of input (written or spoken language, images, and video) as well as the ability to scale the services for a larger userbase, Microsoft's services are worth consideration. There are several possible approaches that can be used in regard to delivering content from Kentico Cloud or Kentico EMS. As mentioned, the effort needed to effectively utilize more advanced features and services offered and to enable the bot to scale with ease will require a bigger investment when designing the bot. Because of this the framework would be a good fit for more advanced or high traffic chatbots.

5.2.2. IBM Watson

Another popular framework is the IBM Watson, which is one of the top frameworks mentioned in the Mindbower's [2017] chatbot survey with 61 percent of respondents preferring it. The IBM Watson API was announced to be made available for developers in 2013 [Upbin 2013]. Since then bots and applications utilizing the features offered by the framework have been launched on various domains. The framework is the result of a long-running investment in the field of artificial intelligence, and it can be said that the IBM backed offering is not going to disappear any time soon. The framework is in active development, as can be seen from the commits to the SDKs that are available on Watson Developer Cloud's GitHub account [GitHub$_f$]. The combined amount of commits for the six available SDKs (.NET, Java, Node.js, JavaScript, Android, Swift, Python, and Unity) is 9857.[22] Regarding documentation, there is a wide array of topics detailing different services offered by the Watson Developer Cloud as well as other services offered by IBM Bluemix platform.[23] Many of the topics also include some examples or videos further explaining the concepts or features.

The framework, like the Bluemix platform in general, has an active community behind it, which offers support. This can be seen, for example, from the StackOverflow posts that are tagged with tags related to Bluemix and Watson, such as "ibm-bluemix" (4587 posts), "ibm-watson-cognitive" (988 posts), "watson-conversation" (454 posts), and "Watson" (442 posts).[24] Additionally, a slack channel and developerWorks forums are offered for posting questions. For example, on the topic "watson" alone, 11255 posts can be found in the forums.[25]

The approach of IBM Watson is to be channel-agnostic, thus, one-click-integrations are not available. However, there is an integration offered through BotKit middleware to help connecting the services to different channels. The framework provides similar services as the Microsoft Bot Framework along with the supporting services, and this makes it similarly a good option for more complex implementations. However, for simpler requirements, the framework

[22] Checked on 19 October 2017 from Watson Developer Cloud GitHub [GitHub$_f$].
[23] For the entire documentation, see [IBM Bluemix].
[24] Number of posts checked on 19 October 2017, see [StackOverflow].
[25] developerWorks forums can be found at [IBM].

might be an overkill, which would require much more time in development than using simpler options would.

5.2.3. Amazon Lex

Amazon Lex is similar in many ways to the other analyzed frameworks. The development of the framework started in the 2010. It has been available for developers as a preview from the end of November 2016, and it was officially launched in the end of April 2017 ([Lardinois 2016; Perez 2017]). The most well know implementation using the Amazon Lex is the previously mentioned Alexa personal assistant. The GitHub repository for Amazon Web Services (AWS) shows that the combined number of commits for the various SDKs (Android, C++, Go, Ruby, JavaScript+Node.js, PHP, iOS, .NET, and Java) available is 20740 (not including a Java SKD v2 Developer preview with 166 commits)[26]. It is also worth mentioning that the SDKs include methods for several services in AWS, not only Lex.

Documentation for Amazon Lex service covers a myriad of topics from initial setup to advanced configuration. As an example of the coverage of the documentation, the Amazon Lex documentation as a pdf is 356 pages long (does not include the separate documentation available for each of the SDKs offered). The documentation is filled with example code snippets as well as full bot examples to showcase the features offered by the framework.

The framework includes integrations for Facebook Messenger, Slack, and Twilio. Lex is tightly integrated with other Amazon AWS services. For example, through the integration with Amazon Lambda, serverless enterprise connectors are available to share data to services such as HubSpot, Marketo, Microsoft Dynamics, Salesforce, Zendesk, and QuickBooks. [Amazon Web Services$_d$; Sanders 2017] Amazon Rekognition can be used to analyze images and detect, for example, objects, scenery, celebrities, and facial features (including expressions to detect sentiment) [Amazon Web Services$_b$].

The framework offers a wide array of services that can be combined to create a flexible chatbot, which uses machine learning and natural language understanding to fulfill the intent of the user. Amazon Lex also provides an easy-to-follow guide on setting up a chatbot with few popular channels. The guide makes it easy to get started with a simple bot. The wide selection of Amazon

[26]Number of commits checked on 21.10.2017. Amazon Web Services GitHub can be found at [GitHub$_a$].

Web Services that can be leveraged alongside Lex make it a versatile option, which is suitable for bots with different requirements.

As can be understood from the analysis in this chapter, the framework can be determined to be suitable for content delivery from services like Kentico Cloud and Kentico EMS. Like the previously analyzed framework, Amazon Lex offers a wide variety of features and an extensive API.

6. Conclusions

The thesis has identified the suitable chatbot options for delivering content from Kentico Cloud and Kentico EMS software. The thesis has moved to this conclusion by first, in Chapter two, introducing and explaining several important terms and concepts, exploring the different ways of categorizing bots, and in order to contextualize the topic, providing a brief history of chatbots. After having established the important terms and concepts as well as the context of the work, Chapter three moved on to familiarize the reader with the different features of the Kentico Cloud and Kentico EMS softwares. After establishing the characteristics of the software relevant for content delivery, Chapter four moved on the take a look at the options available for content delivery. The chapter divided these options into platforms and frameworks and detailed each option's important features concerning content delivery. The Chapter five then delved into analyzing the options' suitability with Kentico products.

In many aspects, the platforms were quite equal with only minor differences between the systems. The frameworks analyzed all proved to have the widest array of features offered, but on the other hand, they would also require most effort for developing/development.

None of the systems can be clearly divided into categories based on the dialog flow as the options allow different kind of flows to be defined. For example, finite-state-based dialog flow control can be achieved with all the analyzed systems. Some options that rely mainly on intents can require more work to limit the conversation to a pre-defined flow than options that allow one to use a visual builder to construct the flow by using blocks. For frame-based approach, some degree of NLP/NLU is required to, for example, properly identify and extract entities from single user utterance and to match the information to the correct slots to allow for the fulfillment of the user intent. Many of the bots combine several strategies for the dialog flow to allow for more flexible and more natural user experience.

Out of the systems analyzed, six out of seven options include some kind of NLU/NLP capabilities out of the box. The only one missing the capabilities is Chatfuel, however, the platform can benefit from the built-in NLP from Facebook Bot platform that it is utilizing. Taking this in to consideration, all the systems can be said to have NLP capabilities. Similarly, all of the systems provide an API for communicating with external applications, although, to varying extents. Some, for example, only offer the ability to query the API for NLP data, while others allow for broadcasting messages to channels. Based on the documentation, the widest APIs are offered by the larger frameworks: Microsoft Bot Framework, IBM Watson, and Amazon Lex.

Regarding offered SDKs, five out of seven options offer at least one SDK for developers. The platforms that do not offer any SDKs are Motion.ai and Chatfuel. In both cases this decision makes sense as the platforms are designed for easily creating chatbots by using a visual builder UI on the service's website. Releasing a SDK would not be in line with their intended use case. The lack of SKDs is, however, a limiting factor as it could mean that additional custom development would be needed if an integration with another application would be required.

Another component of chatbots that was discussed earlier is machine learning. Machine learning was available for total five of the seven options analyzed: Dialogflow, wit.ai, Microsoft Bot Framework, IBM Watson, and Amazon Lex. The last three options offer the most advanced possibilities for utilizing machine learning, though, it is worth mentioning that in most cases the capabilities provided by the other systems probably will suffice, depending of course on the requirements of the implementation. Machine learning does not offer any significant gains in regard to content delivery, thus, concerning the goal of the thesis, this aspect can be excluded.

Most relevant aspect in terms of content delivery are integration possibilities. For internet-connected devices and applications, this would mean, for example, (REST) APIs and web hooks. Both Kentico products offer APIs for integrating with other services. Kentico Cloud also offers web hooks built-in the service. All the analyzed chatbot options included support for webhooks and nearly all of them have a (REST) API.

While writing this thesis, several implementations utilizing chatbots in delivering content from Kentico Cloud and Kentico EMS emerged, a list of implementations is provided in Appendix 1. The implementations further prove the conclusions reached in the thesis

References

Martin Abadi et al. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv:1603.04467 [cs.DC], 16 March, 1–19.

Sameera A. Abdul-Kader, and John Woods. 2015. Survey on chatbot design techniques in speech conversation systems. *IJACSA 6*, 7, 72–80.

Bayan AbuShawar, and Eric Atwell. 2016. Usefulness, localizability, humanness, and language-benefit: additional evaluation criteria for natural language dialogue systems. *Int J Speech Technol,*19, 1, 373–383.

Amazon Developer. Slot type reference. https://developer.amazon.com/docs/custom-skills/slot-type-reference.html (23.10.2017).

Amazon Web Services[a]. Amazon Lex console. https://console.aws.amazon.com/lex/ (23.10.2017).

Amazon Web Services[b]. Amazon Rekognition. https://aws.amazon.com/rekognition/ (23.10.2017).

Amazon Web Services[c]. Exporting amazon Lex bots. http://docs.aws.amazon.com/lex/latest/dg/export.html (23.10.2017).

Amazon Web Services[d]. What is Amazon Lex? http://docs.aws.amazon.com/lex/latest/dg/what-is.html (23.10.2017).

Artificial Intelligence Foundation. A.I. Foundation chat bot survey. http://www.alicebot.org/aimlbots.html (22.10.2017).

Markus M. Berg. 2014. *Modelling of Natural Dialogues in the Context of Speech-based Information and Control Systems*. Ph.D. Dissertation, der Technischen Fakultät, der Christian-Albrechts-Universität zu Kiel.

Markus M. Berg, and Antje Düsterhöft. 2010. Website interaction with text-based natural language dialog systems. In: 7. *Wismarer Wirtschaftsinformatiktage.*

Jeff Bertolucci. 2016. Chat bots 101: A primer for app developers. *IBM.* https://www.ibm.com/blogs/watson/2016/10/chat-bots-101-primer-app-developers/ (22.10.2017).

Kim Brandl, and Robert Standeref. 2017. Azure Bot Service. *Microsoft*. https://docs.microsoft.com/en-us/bot-framework/azure-bot-service-overview (23.10.2017).

Kim Brandl, Denise Mak, Robert Standefer, Den Delimarsky, Jayme M. Perlman. 2017. Connect a bot to channels. *Microsoft*. https://docs.microsoft.com/en-us/bot-framework/portal-configure-channels (23.10.2017).

Chatfuel$_a$. Build a Facebook bot without coding. https://chatfuel.com/ (23.10.2017).

Chatfuel$_b$. Chatbot-building eBook from Chatfuel & HubSpot. https://blog.chatfuel.com/chatbot-building-ebook-from-chatfuel-hubspot/ (23.10.2017).

Chatfuel$_c$. Documentation. http://docs.chatfuel.com/ (23.10.2017).

Chatfuel$_d$. Free tool - CheckBox plugin button generator. https://community.chatfuel.com/t/free-tool-checkbox-plugin-button-generator/13102 (23.10.2017).

Chatfuel$_e$. Plugins. http://docs.chatfuel.com/plugins (23.10.2017).

Chatfuel$_f$. User input. http://docs.chatfuel.com/plugins/plugin-documentation/user-input (23.10.2017).

Dialogflow$_a$. Agents .https://dialogflow.com/docs/agents (22.10.2017).

Dialogflow$_b$. Basics. https://dialogflow.com/docs/getting-started/basics (23.10.2017).

Dialogflow$_c$. Contexts. https://dialogflow.com/docs/contexts (23.10.2017).

Dialogflow$_d$. Entities. https://dialogflow.com/docs/entities (21.10.2017).

Dialogflow$_e$. Integrations. https://dialogflow.com/docs/integrations/ (22.10.2017).

Dialogflow$_f$. Intents. https://dialogflow.com/docs/intents (21.10.2017).

Dialogflow$_g$. Introducing Dialogflow, the new name for API.AI. https://blog.dialogflow.com/post/apiai-new-name-dialogflow-new-features/ (23.10.2017).

Dialogflow$_h$. Languages. https://dialogflow.com/docs/reference/language (23.10.2017).

Dialogflow$_i$. Machine learning. https://dialogflow.com/docs/machine-learning (23.10.2017).

Dialogflow$_j$. Multi-language agents. https://dialogflow.com/docs/multi-language (23.10.2017).

Dialogflow$_k$.. SDKs. https://dialogflow.com/docs/sdks (22.10.2017).

Facebook for Developers. Natural language processing. https://developers.facebook.com/docs/messenger-platform/built-in-nlp (23.10.2017).

Facebook Newsroom. 2016. Messenger platform at F8. https://newsroom.fb.com/news/2016/04/messenger-platform-at-f8/ (21.10.2017).

Luca Friedrich. JSON API. *Chatfuel.* http://docs.chatfuel.com/plugins/plugin-documentation/json-api (22.10.2017).

GitHuba. Amazon web services. https://github.com/aws (23.10.2017).

GitHubb. Kentico. https://github.com/kentico (23.10.2017).

GitHubc. Microsoft/BotBuilder. https://github.com/Microsoft/BotBuilder/pulse/monthly (24.10.2017).

GitHubd. MotionAI/nodejs-samples. https://github.com/MotionAI/nodejs-samples (22.10.2017).

GitHube. watson-developer-cloud/botkit-middleware. https://github.com/watson-developer-cloud/botkit-middleware (23.10.2017).

GitHubf. Watson developer cloud. https://github.com/watson-developer-cloud (23.10.2017).

Carol Hanna, and Denise Mak. 2017. Label suggested utterances. https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/label-suggested-utterances (23.10.2017).

IBM. Watson developerWorks answers. https://developer.ibm.com/answers/topics/watson.html (23.10.2017).

IBM Bluemix. 2017a. Building a bot with botkit. https://console.bluemix.net/docs/services/conversation/integrations.html#building-a-bot-with-botkit (23.10.2017).

IBM Bluemix. 2017b. Building a dialog. https://console.bluemix.net/docs/services/conversation/dialog-build.html#dialog-build (23.10.2017).

IBM Bluemix. Get started by deploying your first app. https://console.bluemix.net/docs/ (23.10.2017).

IBM Watson. Watson products and services. https://www.ibm.com/watson/products-services/ (23.10.2017).

Kristiina Jokinen, and Michael McTear. 2010. *Spoken Dialogue Systems*. Morgan & Claypool Publishers.

Daniel Jurafsky, and James H. Martin. 2000. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR.

Kentico[a]. All features. https://www.kentico.com/product/all-features (23.10.2017).

Kentico[b]. Creating new pages. https://docs.kentico.com/k10/managing-website-content/working-with-pages/creating-new-pages#Creatingnewpages-CreatingnewpagesinthePagesapplication (23.10.2017).

Kentico[c]. Extensibility and API. https://www.kentico.com/product/all-features/development/extensibility (23.102017).

Kentico[d]. Kentico 10 documentation. https://docs.kentico.com/k10 (23.10.2017).

Kentico[e]. Overview. https://www.kentico.com/product/overview (23.10.2017).

Kentico[f]. Server and hosting requirements. https://docs.kentico.com/k10/installation/server-and-hosting-requirements (23.10.2017).

Kentico Cloud[a]. API Reference. https://developer.kenticocloud.com/reference#api-introduction (23.10.2017).

Kentico Cloud[b]. Authoring experience and collaboration. https://help.kenticocloud.com/authoring-experience-and-collaboration#collaboration (23.10.2017).

Kentico Cloud[c]. Content delivery. https://kenticocloud.com/content-delivery (23.10.2017).

Kentico Cloud[d]. Content management. https://kenticocloud.com/content-management (23.10.2017).

Kentico Cloud[e]. Delivering content. https://developer.kenticocloud.com/docs/delivering-content (23.10.2017).

Kentico Cloud[f]. Kentico cloud roadmap. https://kenticocloud.com/roadmap (23.10.2017).

Tina Klüwer. 2011. From Chatbots to Dialog Systems. In: Diana Perez-Marin and Ismael Pascual-Nieto (eds.), *Conversational Agents and Natural Language Interaction*. IGI Global, 1–22.

Frederic Lardinois. 2016. Amazon launches Amazon AI to bring its machine learning smarts to developers. *Tech Crunch*. https://techcrunch.com/2016/11/30/amazon-launches-amazon-ai-to-bring-its-machine-learning-smarts-to-developers/ (23.10.2017).

Denise Mak, Robert Standefer, Duc Cash Vo, Den Delimarsky, and Jayme M. Perlman. 2017. Debug bots with the Bot Framework Emulator. *Microsoft*. https://docs.microsoft.com/en-us/bot-framework/debug-bots-emulator (23.10.2017).

Microsoft. 2016. Microsoft Bot Framework. https://blog.botframework.com/2016/03/30/botframework/ (24.10.2017).

Microsoft Azure. Cognitive services directory. https://azure.microsoft.com/en-gb/services/cognitive-services/directory/ (23.10.2017).

Mindbowser 2017. Global chatbot trends report – 2017. http://mindbowser.com/chatbot-market-survey-2017/ (23.10.2017).

Motion.ai$_a$. Connections. https://docs.motion.ai/docs/module-connections (23.10.2017).

Motion.ai$_b$. Custom/API. https://docs.motion.ai/docs/api (23.10.2017).

Motion.ai$_c$. Dashboard. https://dashboard.motion.ai/ (23.10.2017).

Motion.ai$_d$. FAQs. https://docs.motion.ai/v1.0/docs/frequently-asked-questions (23.10.2017).

Motion.ai$_e$. Modules. https://docs.motion.ai/docs/what-are-modules (22.10.2017).

Motion.ai$_f$. NLP Basics - Enabling the NLP engine. *Youtube*. https://www.youtube.com/watch?v=YJ4NV25zT6g&list=PLCo77wGSKVeOnnA0zmVhiOL6BHOdG5mNi (23.10.2017).

Motion.ai$_g$. NLP documentation. https://docs.motion.ai/discuss/595ac34e2e3626002b8fdb87 (23.10.2017).

Pandorabots. Getting started with chatbot development. http://docs.pandorabots.com/tutorials/getting-started/ (22.10.2017).

Sarah Perez. 2017. Amazon Lex, the technology behind Alexa, opens up to developers. *Tech Crunch*. https://techcrunch.com/2017/04/20/amazon-lex-the-technology-behind-alexa-opens-up-to-developers/ (23.10.2017).

PR Newswire. 2014. Teens use voice search most, even in bathroom, Google's mobile voice study finds. https://www.prnewswire.com/news-releases/teens-use-voice-search-most-even-in-bathroom-googles-mobile-voice-study-finds-279106351.html (21.10.2017).

Nicole Radziwill and Morgan Benton. 2017. Evaluating quality of chatbots and intelligent conversational agents. ArXiv:1704.04579 [cs.CY], June, 21 pages.

Rick Ramos. 2017. The future of enterprise chatbots. *Venturebeat*. https://venturebeat.com/2017/07/20/the-future-of-enterprise-chatbots/ (21.10.2017).

James Sanders. 2017. Amazon Lex: The smart person's guide. Tech Republic. https://www.techrepublic.com/article/amazon-lex-the-smart-persons-guide/ (23.10.2017).

Cruce Saunders. 2017. [A] slide deck: Engineering content for bots, AI, and marketing automation. https://simplea.com/Publications/Decks/Engineering-Content-for-Bots-AI (21.10.2017).

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. arXiv:1506.06714 [cs.CL], 22 June, 196–205.

StackOverflow. Newest questions. https://stackoverflow.com/questions (23.10.2017).

Robert Standefer, Duc Cash Vo, Denise Mak, and Den Delimarsky 2017a. Bot Builder SDK for .NET samples. *Microsoft*. https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-samples (23.10.2017).

Robert Standefer, Duc Cash Vo, Denise Mak, and Den Delimarsky 2017b. Bot Builder SDK for Node.js samples. *Microsoft*. https://docs.microsoft.com/en-us/bot-framework/nodejs/bot-builder-nodejs-samples (23.10.2017).

Juraj Uhlar[a]. Content type elements reference. *Kentico Cloud*. https://help.kenticocloud.com/define-content-structure/content-elements/content-type-elements-reference (23.10.2017).

Juraj Uhlar[b]. Creating and deleting content types. *Kentico Cloud*. https://help.kenticocloud.com/define-content-structure/structure/creating-and-deleting-content-types (23.10.2017).

Bruce Upbin 2013. IBM opens up its Watson cognitive computer for developers everywhere. *Forbes*.

https://www.forbes.com/sites/bruceupbin/2013/11/14/ibm-opens-up-watson-as-a-web-service/#1b19a08377ef (24.10.2017).

Oriol Vinyals, and Quoc Le. 2015. A Neural Conversational Model. arXiv:1506.05869 [cs.CL], 22 June, 8 pages.

Joseph Weizenbaum. 1966. ELIZA - A computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9, 1, 36–45.

Wikipedia$_a$. Content delivery network. https://en.wikipedia.org/wiki/Content_delivery_network (23.10.2017).

Wikipedia$_b$. Interactive voice response. https://en.wikipedia.org/wiki/Interactive_voice_response (21.10.2017).

Wit.ai$_a$. Getting Started with wit.ai. https://wit.ai/docs (23.10.2017).

Wit.ai$_b$. HTTP API Reference. https://wit.ai/docs/http/20170307 (23.10.2017).

Wit.ai$_c$. 2017. Launching built-in NLP for Messenger and sunsetting bot engine (beta). https://wit.ai/blog/2017/07/27/sunsetting-stories (23.10.2017).

Wit.ai$_d$. Recipes for apps you can talk to. https://wit.ai/docs/recipes (23.10.2017).

Wit.aie. Which entity should I use?. https://wit.ai/docs/recipes#which-entity-should-i-use (22.10.2017).

Appendix 1: List of example implementations for chatbots utilizing Kentico Cloud or Kentico EMS

Brian McKeiver, Kentico EMS, E-commerce bot	http://www.mcbeev.com/Blog/February-2017/Building-a-Kentico-E-Commerce-Chat-Bot
Michael Kinkaid, Connecting Kentico Cloud, Chat Bots and Google Home	https://www.youtube.com/watch?v=YlpGWCizdvU
Bryan Soltis, Using Slack webhooks with Kentico Cloud	https://kenticocloud.com/blog/using-slack-webhooks-with-kentico-cloud
Bryan Soltis, Using an Azure Function Webhook with Kentico Cloud	https://kenticocloud.com/blog/using-an-azure-function-webhook-with-kentico-cloud
Bryan Soltis, Powering Alexa with Kentico Cloud	https://kenticocloud.com/blog/powering-alexa-with-kentico-cloud
Bryan Soltis and, Building Applications Using Microservices and Azure – Part 1	https://kenticocloud.com/blog/developing-apps-using-microservices-and-azure-1
Ondrej Fridrich, Building Applications Using Microservices and Azure – Part 2	https://kenticocloud.com/blog/building-apps-using-microservices-and-azure-2