# Desirable Features of a Chatbot-building Platform

Saurabh Srivastava
*Department of Computer Science & Engineering*
*Indian Institute of Technology*
Kanpur, India
ssri@iitk.ac.in

T.V. Prabhakar
*Department of Computer Science & Engineering*
*Indian Institute of Technology*
Kanpur, India
tvp@iitk.ac.in

***Abstract*—There is a visible eagerness in the business community to integrate *chatbots* with their websites and mobile apps. They provide a humanised interface to information and can serve as digital assistants that can perform tasks on behalf of an individual. There are many commercial platforms which provide interfaces to build these chatbots. They are used by both professional software developers as well as people from non-IT backgrounds. Based on our experiences with three popular chatbot-building platforms - Google Dialogflow, IBM Watson Assistant and Amazon Lex, we present a list of desirable features that these platforms should exhibit in order to cater to their mixed user base. We also rate the availability and ease of use of these features on the current versions of these platforms.**

## I. Background

Chatbots are deployed to serve use cases in many domains such as Customer Support, E-commerce, News Services, Healthcare etc. [1]. They are becoming extremely popular with businesses, and are expected to become ubiquitous in near future [2]. There are dozens of commercial platforms today which allow users to create and deploy chatbots of varying capabilities. Some platforms are specifically targeting users from non-IT backgrounds, boasting a "coding-free" experience to build the chatbots [3]. However, such chatbots are usually limited to performing common tasks that the platform builders envisaged beforehand. This is why most of the platforms also provide enough features to accommodate professional software developers. It presents an interesting trade-off - keeping the development dashboard simple to avoid intimidating a user from the non-IT background, and at the same time, provide enough flexibility to software professionals to build sophisticated chatbots that suit their requirements.

### A. Chatbot-building Platforms

Almost all the chatbot-building platforms today expect the chatbots to be defined in a particular pattern. We term this pattern as the *Contextual Reactive* pattern [4]. The idea is to define a *context* - e.g. "the customer wants to order a product" or "the customer wants to report a fraudulent transaction", and the *reaction* to this context - e.g. "invoke the purchase business function" or "transfer the chat to a human". The context is defined by declaring a set of Intents and Entities and supplying some sample utterances of the customer. Intents signify the different types of queries that the chatbot caters to (e.g. "Sales Query" or "Support Query"), while Entities refer to instances of particular details required to process a query (e.g. "Name of the product" or "Error code"). The reaction could be defined using static responses such as "Sure, I'll process your order", or it can be redirected to a complex processing pipeline involving the execution of one or more business functions [5]. We have previously explored these elements of a chatbot's definition [6]. In this work, we concentrate more on the features of the dashboard of these platforms, although we do discuss, in brief, other modes that the platform should provide (e.g. APIs) to provide development flexibility. We also provide a *relative comparison* between three commercial platforms - Google Dialogflow [7], IBM Watson Assistant [8] and Amazon Lex [9] - over their offerings of these features.

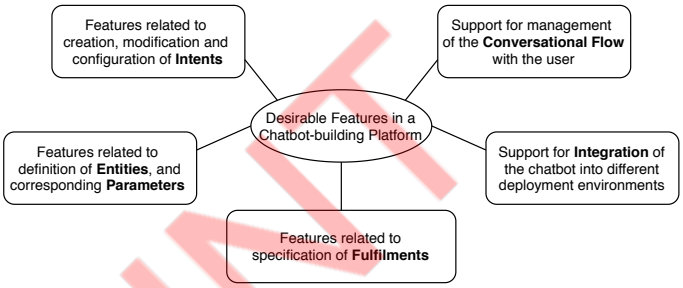


Fig. 1. Top-level view of the desired features

### B. Related Work

There is no dearth of articles which compare different chatbot platforms ([10] [11] [12] [13] etc.). However, they usually cover the platforms from the perspective of a professional software developer. Some work has also been initiated to view these platforms from a business perspective [14] [15]. We too have investigated the features provided by a chatbot-building platform in one of our previous work [16], though it was from the perspective of a software architect. The current work attempts to condense existing knowledge about these platforms into an abstract feature-list, which can be helpful for developers of novel platforms. This work can also act as a cheat-sheet for chatbot builders to evaluate the utility of a platform for their use case.

## II. Feature List

The Feature List that we have compiled is hierarchical in nature. These features are organised in a 3-tier hierarchy. Figure 1 shows the top-level view of these features. Each

TABLE I

DESIRED FEATURES OF A CHATBOT-BUILDING PLATFORM AND EVALUATION OF THEIR RELATIVE SUPPORT ON THREE COMMERCIAL PLATFORMS

| Desired Platform Features | Dialogflow | Watson Assistant | Lex |
|---|---|---|---|
| ***Intent Management Features*** | ***1*** | ***0.9583*** | ***0.8177*** |
| *Create Intents* | *1* | *1* | *0.9375* |
| Add Parameters | 1 | 1 | 1 |
| Add training examples | 1 | 1 | 0.75 (A) |
| Tag Parameter occurrences in training examples | 1 | 1 | 1 (A) |
| Map Parameter occurrences to specific values | 1 | 1 | 0 (A) |
| *Update Intents* | *1* | *1* | *1* |
| Provide new training examples | 1 | 1 | 1 |
| Edit existing training examples | 1 | 1 | 1 |
| Remove existing training examples | 1 | 1 | 1 |
| *Delete Intents* | *1* | *1* | *1* |
| *Default Intent* | *1* | *0.8333* | *0.3333* |
| Provide default response | 1 | 1 | 1 |
| Set a minimum confidence threshold to trigger any Intent | 1 | 0.75 (B) | 0 (B) |
| Provide counterexamples – to trigger Default Intent explicitly | 1 | 0.75 (C) | 0 (C) |
| ***Entity and Parameter Management Features*** | ***1*** | ***0.9167*** | ***0.8833*** |
| *Create Parameters* | *1* | *0.75* | *1* |
| Provide possible values for the Parameter | 1 | 1 | 1 |
| Provide synonymous occurrences for provided values | 1 | 1 | 1 |
| Allow the extension of possible values list automatically | 1 | 1 | 1 |
| Disallow extension of possible values list automatically | 1 | 0 (D) | 1 |
| *Update Parameters* | *1* | *1* | *0.65* |
| Add more possible values for the parameter | 1 | 1 | 0.75 (E) |
| Edit a provided value | 1 | 1 | 0.75 (E) |
| Change synonymous occurrences | 1 | 1 | 0.25 (E) |
| Delete synonymous occurrences | 1 | 1 | 0.75 (E) |
| Remove existing values for the parameter | 1 | 1 | 0.75 (E) |
| *Delete Parameters* | *1* | *1* | *1* |
| ***Fulfilment Management Features*** | ***0.8417*** | ***0.9833*** | ***0.7667*** |
| *Slot filling* | *1* | *1* | *1* |
| Provide prompts to receive inputs from the user during conversation, for filling a required Slot | 1 | 1 | 1 |
| Show options (one or more buttons for the user to click, out of a limited set) | 1 | 1 | 1 |
| *Provide a static response for some queries* | *0.875* | *1* | *0.75* |
| Textual Response | 1 | 1 | 0.75 (F) |
| Multimedia Response (Image/Audio/Video etc.) | 0.75 (G) | 1 | 0.75 (F) |
| *Provide a dynamically crafted response for some queries* | *0.3333* | *0.9167* | *0.3333* |
| Allow the response to have placeholders for Slot values and Context variables | 1 | 1 | 1 |
| Allow the response to have placeholders for secondary information, e.g. Intent classification confidence or computed values | 0 (H) | 1 | 0 (H) |
| Dynamic response based on variable values (e.g. using conditional operators or `if-else` ladders) | 0 (H) | 0.75 (H) | 0 (H) |
| *Trigger external events* | *1* | *1* | *1* |
| *Control Slot values interpretation* | *1* | *1* | *0.75* |
| Get the supplied value | 1 | 1 | 0.75 (I) |
| Get the reference value | 1 | 1 | 0.75 (I) |
| ***Integration Features*** | ***0.75*** | ***0.9375*** | ***0.6875*** |
| *Send requests to a public URL and expect a response back* | *0.75* (J) | *1* (J) | *0.25* (J) |
| *Send request to a restricted source (a URL with domain restrictions or a function that can be invoked internally by the platform) and expect a response back* | *0.25* (K) | *1* (K) | *1* (K) |
| *Interfaces to integrate with existing platforms (e.g. Facebook Messenger, Slack, Telegram etc.)* | *1* (L) | *0.75* (L) | *0.5* (L) |
| *API to create, modify, train, use and delete the Chatbot (e.g. REST APIs)* | *1* (M) | *1* (M) | *1* (M) |
| ***Conversation Flow Management Features*** | ***0.5625*** | ***0.6875*** | ***0.3125*** |
| *Pass contextual information from the currently triggered Intent to any subsequently triggered Intents in the conversation flow* | *1* | *1* | *0.5* (N) |
| *Handle follow-ups (e.g. Confirmations such as Yes or No, Next or Previous etc.)* | *1* (O) | *0.5* (O) | (O) |
| *Handle digressions* | *0.25* (P) | *1* (P) | *0* (P) |
| *Process multiple-Intents queries* | *0* (Q) | *0.25* (Q) | *0* (Q) |

TABLE II
BRIEF EXPLANATIONS FOR SELECTED ENTRIES IN TABLE I

| | |
|---|---|
| (A) | Lex only expect placeholders instead of actual values, e.g. a training example in Lex looks like "Book a ticket from *$source* to *$destination*", as compared to say a tagged example in Dialogflow or Watson Assistant, e.g. "Book a ticket from `Delhi:`*source* to `Mumbai:`*destination*". |
| (B) | In Watson Assistant, although an explicit value cannot be specified, an if condition in the Dialog Tree can handle it. No direct or indirect mechanism found in Lex for doing the same. |
| (C) | In Watson Assistant, it cannot be done explicitly but can be done by tagging an example `irrelevant` in the dashboard or passing it indirectly by restoring a skill from a JSON file. No direct or indirect mechanism found in Lex for doing the same. |
| (D) | Applicable to Contextual Entities - Watson Assistant tries to match values with the context, and there is no way to stop it from adding a new value to the set of existing, pre-defined values. |
| (E) | In Lex, updating a Parameter is not possible directly. The Parameter must first be added to an Intent, only then can it be edited. The chatbot developer may find the interface for adding/modifying synonyms confusing. For example, if an entity value had only one synonym, deletion of the synonym is not allowed in the editing popup. The workaround is to delete the value itself, and add it again (without any synonym). |
| (F) | Lex does not provide a way to provide a static response straightaway to the user. A fulfilment step must be completed (either by invoking a Lambda function or returning the parameters back to the client). Only after the completion of the fulfilment, can a static response be shown. |
| (G) | Dialogflow allows rich responses for a set of target platforms (e.g. Facebook Messenger or Slack). A simple image response cannot be sent. It must be configured differently for different platforms. |
| (H) | In Dialogflow, static placeholders can be added in the response, e.g. "Your `$product` has been shipped", but no mechanism for implementing conditions. In Watson Assistant, simple conditions can be handled using the conditional (`?`) operator, while the Dialog Tree can be used for complex use cases. In Lex, a separate Lambda function can be configured for "initialization and validation", but no mechanism for implementing conditions. |
| (I) | In Lex, if the chosen option is "supplied values", synonyms are ignored. It might be preferable to get the reference value whenever supplied value matches the reference value (or provided synonyms) exactly, and get the supplied value as fallback. In Watson Assistant, the reference value is provided by default (with expression `@entity`), whereas the supplied value can be retrieved by adding the literal property (with expression `@entity.literal`). Dialogflow has a similar case. The property for supplied value is `original` (instead of `literal`). |
| (J) | Dialogflow has a timeout of `5 seconds`, whereas Watson Assistant has a timeout of `8 seconds`. Lex can only redirect flow to an external caller, if this is how the conversation was initiated. The only other option is to invoke a Lambda function. |
| (K) | Dialogflow allows writing inline fulfilment code, which is deployed as a Google Cloud Function. Only Node.js can be used for this purpose. Watson Assistant can connect to any IBM Cloud Function, and Lex can connect to any AWS Lambda function, which in turn can be implemented in any language (some languages are supported natively, while others can be supported via docker images and runtime APIs). |
| (L) | Dialogflow: Telephony - Dialogflow Phone, Avaya, SignalWire, Voximplant, AudioCodecs, Genesys Cloud; Text based - Web Demo Widger, Dialogflow Messenger, Facebook Messenger, Slack, Viber, Twitter, Twilio IP, Twilio SMS, Skype, Telegram, Kik, Line, Cisco Sparc, Amazon Alexa<br>Watson Assistant: Webchat Widget, Salesforce, Zendesk, Intercom (Premium Accounts) and Facebook Messenger, Slack, Voice Agent (Telephony)<br>Lex: Facebook Messenger, Kik, Slack, Twilio SMS |
| (M) | Dialogflow added support for it in the June 13, 2019 release and updated it in the February 20, 2020 release. Watson Assistant provides support for creation and modification of skills through only v1 API [17]. The v2 API only provides methods to interact with the assistant. A "workspace" in v1 API is equivalent to a "skill" in v2 API [18]. Lex provides a well-documented user guide to access its AWS CLI for Lex related tasks [19]. |
| (N) | In Lex, while context variables can be used across Intents, they cannot be set in the UI. They must be set in the responses sent by Lambda Functions. |
| (O) | Dialogflow offers many "follow-up Intents" such as `Yes`, `No`, `Previous`, `Next`, `Cancel` etc. Lex provides Intents through the Alexa Skill set (except `Yes` and `No` Intents). Watson Assistant doesn't provide such features, but the same can be done via context variables in the Dialog Tree. |
| (P) | Watson Assistant provides a dedicated feature to handle digressions. In Dialogflow, some cases of digressions may be handled through a complex setting and unsetting of input and output context variables [20]. In Lex, no specific feature is provided to handle digressions, nor do they advertise if setting different context variables will change the behaviour of the Intent matching algorithm. |
| (Q) | The Dialog Tree in Watson Assistant may be able to handle multiple-intent queries, but getting it right is fairly difficult [21]. Dialogflow and Lex does not have any visible mechanism to handle multiple-intent queries. |



category shown in the figure is divided into subcategories, which in turn are divided further into finer technical details. The leftmost column of Table I shows the complete feature list. The indentation separates the different levels.

### A. Using the Feature List

We suggest that the users familiarize themselves with the development paradigm, including the Contextual Reactive pattern [4] for defining the chatbots. Picking one or more candidate platforms for developing the chatbot is also essential. The next step should be investigating the advertised features of the chosen platform(s) with respect to the top-level categories shown in Figure 1. For readers from non-IT background, we suggest having a look at the second tier of the list shown in Table II, with exception to the last two categories (i.e. ***Integration Features*** and ***Conversation Flow Management Features***), which are more useful for professional software developers. The third tier covers technical details and has more utility for experienced chatbot builders. The list can also be useful for software architects for evaluating the support of one or more candidate platforms from a Quality Attributes perspective, as discussed in [16].

### B. Case Studies

During our previous work [16], we investigated three chatbot platforms - Google Dialogflow [7], IBM Watson Assistant [8] and Amazon Lex [9] in significant detail. We, therefore, decided to rate the support for the discussed features on these platforms. The criteria we used for ranking is discussed below:

- We evaluated the platforms for all the features at the third tier. We rated the platforms on a scale of `0` to `1` for each feature.
- If the feature was not available on the platform, we assigned a value of `0`. If the feature was available and easy to use, we assigned a value of `1`. For cases where a feature was either partially present, or, its usage was not easy, we assigned a value between `0` and `1`, with three different intermediate levels - `0.25`, `0.5` and `0.75`.

- The intermediate values, when used, were relative in nature. For example, for the feature *Interfaces to integrate with existing platforms*, a value of `0.75` for Watson Assistant and `0.5` for Lex are relative to the value `1` for Dialogflow. This means that for the said feature, Dialogflow's support is better than Watson Assistant, which, in turn, is better than Lex.
- We also calculated accumulated scores of the platform on higher levels of the hierarchy, by taking an equal-weightage weighted sum of the constituent scores. For instance, the score for *Default Intents* for Watson Assistant is calculated by adding the support for the three technical details and dividing it by `3` (i.e. `(1 + 0.75 + 0.75) / 3`).
- For each case where we assign a value other than `1`, we provide a short explanation in Table II. The respective entries of explanation in Table II are shown in corresponding cells of Table I.

The Feature List that we compiled is based on the common tasks that a chatbot builder is supposed to perform. It must be noted that all three platforms can be used to build professional chatbots. However, what may differ from one platform to the other is the amount of effort that is required to do so. For first-time chatbot builders, this list also provides subtle hints to the tasks that they may have to undertake for building the chatbot.

## III. Conclusion and Future Work

Chatbots are one of the most popular humanised interfaces to systems today. Many commercial chatbot-building platforms have cropped up in the past few years. These platforms provide dashboards to build these chatbots with no or minimal amount of coding. In the current work, we presented a list of desired features that these platforms should expose. We also analysed the current versions of three popular platforms - Google Dialogflow, IBM Watson Assistant and Amazon Lex - and presented a relative comparison between the support that they provide for each feature.

Since chatbots are a relative novelty for software environments, the chatbot-building platforms are also evolving constantly. This means that there may be several exciting features which may be in pipeline currently. Thus, we would like to keep updating this list in future, to include any other additions on these platforms which can make the whole process of building a chatbot easier. Also, for the current work we chose three popular and relatively stable platforms for analysis. In future, we would also like to explore other popular chatbot building platforms such as Chatfuel [22], Gupshup [23], Botsify [24] etc.

## References


[1] "The Best Chatbot Examples and Awesome Chatbot Ideas That You Can Borrow," https://www.tidio.com/blog/chatbot-examples/, accessed: 2019-07-20.

[2] "80% of businesses want chatbots by 2020." https://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12, accessed: 2019-07-20.

[3] "7 best chatbot building platforms that require no coding," https://roboticsbiz.com/6-best-chatbot-building-platforms-that-require-no-coding/, accessed: 2019-07-20.

[4] S. Kalra, T. V. Prabhakar, and S. Srivastava, "Contextual Reactive Pattern on Chatbot-building Platforms," in *Proceedings of the 25th European Conference on Pattern Languages of Programs*, 2020.

[5] S. Srivastava and T. V. Prabhakar, "A Reference Architecture for Applications with Conversational Components," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2019, pp. 1–5.

[6] ——, "Intent Sets: Architectural Choices for Building Practical Chatbots," in *Proceedings of the 2020 12th International Conference on Computer and Automation Engineering*, 2020, pp. 194–199.

[7] Google LLC, "Dialogflow," 2010, [Accessed: 2019-07-20]. [Online]. Available: https://dialogflow.com/

[8] IBM, "Watson Assistant," 2016, [Accessed: 2019-07-20]. [Online]. Available: https://www.ibm.com/cloud/watson-assistant/

[9] Amazon.com Inc., "Amazon Lex – Build Conversation Bots," 2016, [Accessed: 2019-07-20]. [Online]. Available: https://aws.amazon.com/lex/

[10] "Chatbot Platforms: A Comparative Table," https://chatbotsjournal.com/25-chatbot-platforms-a-comparative-table-aeefc932eaff, accessed: 2019-07-20.

[11] "2020 Chatbot Platform Comparison Reviews," https://www.ometrics.com/blog/chatbot-platform-comparison-reviews/, accessed: 2019-07-20.

[12] "Top 175 chatbot platforms of 2020: In-Depth Guide," https://aimultiple.com/chatbot-platform, accessed: 2019-07-20.

[13] "Chatbot Platform Comparison," https://botpress.com/blog/chatbot-platform-comparison, accessed: 2019-07-20.

[14] B. Galitsky, *Developing Enterprise Chatbots: Learning Linguistic Structures*. Springer, 2019.

[15] P. Kostelník, I. Pisařovic, M. Muroň, F. Dařena, D. Procházka *et al.*, "Chatbots for Enterprises: Outlook," *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, vol. 67, no. 6, pp. 1541–1550, 2019.

[16] S. Srivastava and T. V. Prabhakar, "Hospitality of Chatbot Building Platforms," in *Proceedings of the 2nd ACM SIGSOFT International Workshop on Software Qualities and Their Dependencies*, 2019, pp. 12–19.

[17] "Watson Assistant v2 - IBM Cloud API Docs," https://cloud.ibm.com/apidocs/assistant/assistant-v2, accessed: 2019-07-20, See the first "Tip".

[18] "Watson Assistant API overview," https://cloud.ibm.com/docs/assistant?topic=assistant-api-overview, accessed: 2019-07-20, See the second "Note".

[19] "lex-models - AWS CLI 1.18.101 Command Reference," https://docs.aws.amazon.com/cli/latest/reference/lex-models/index.html, accessed: 2019-07-20.

[20] "Input and output contexts — Dialogflow Documentation — Google Cloud," https://cloud.google.com/dialogflow/docs/contexts-input-output, accessed: 2019-07-20.

[21] "Handling Multi-Intent Questions in Watson Assistant," https://medium.com/ibm-watson/handling-multi-intent-questions-in-watson-assistant-ccd0c6ea21e1, accessed: 2019-07-20.

[22] "Chatfuel," https://chatfuel.com/, accessed: 2019-07-20.

[23] "Gupshup.io," https://www.gupshup.io/developer/home, accessed: 2019-07-20.

[24] "Botsify," https://botsify.com/, accessed: 2019-07-20.