Are the Tools up to the Task? An evaluation of commercial dialog tools in developing conversational enterprise-grade dialog systems

Marie Meteer, Meghan Hickey, Ellen Eide Kislal, David Nahamoo, Carmi Rothberg
Pryon, Inc.
mmeteer, mhickey, ekislal, dnahamoo, crothberg@pryoninc.com

Abstract

There has been a significant investment in dialog systems (tools and runtime) for building conversational systems by major companies including Google, IBM, Microsoft, and Amazon. The question remains whether these tools are up to the task of building conversational, task-oriented dialog applications at the enterprise level. In our company, we are exploring and comparing several toolsets in an effort to determine their strengths and weaknesses in meeting our goals for dialog system development: accuracy, time to market, ease of replicating and extending applications, and efficiency and ease of use by developers. In this paper, we provide both quantitative and qualitative results in three main areas: natural language understanding, dialog, and text generation. While existing toolsets were all incomplete, we hope this paper will provide a roadmap of where they need to go to meet the goal of building effective dialog systems.

1 Introduction

With the explosion of smart devices and the significant improvement in speech recognition over the past few years, the demand for intelligent, conversational dialog systems is rising quickly. At our company we are working to meet that demand in the enterprise market to provide secure and natural means for accessing information and improving business processes. Essential to meeting this demand is getting dialog systems into the hands of customers as quickly as possible while ensuring accurate interpretation of utterances as well as conversational means of handling ambiguity, error, and out of domain/out of scope utterances.

In this paper we explore whether the dialog tools available from a number of companies are up to the task of creating complete systems efficiently. We provide both quantitative and qualitative results in three main areas:

- Natural language understanding: intent and entity accuracy, out-of-domain and out-of-scope identification, and anaphora and coreference resolution.
- Dialog management: frame-based and contextual dialog control, dialog structure, and digression.
- Response generation: text generation, error correction and clarification.

We provide results and examples across multiple domains to illustrate the challenges in developing a truly conversational dialog system. As we will show, there is considerable progress in accuracy of intents and entities in constrained domains across all of the toolsets. However, capturing subtle differences in in-scope vs. out-of-scope utterances is still difficult, as is partial understanding, which is important for clarification. For dialog management, all of the tools offer some ability to use frames, however they differ in how robust they are to context. Similarly, for error correction and clarification, the ability to react based on context is limited. Finally, in text generation, we found there was no support beyond template-based generation, which impacts not only naturalness, but the ability to interpret a user's follow-on utterance.

There has been other work comparing just the accuracy of these tools (e.g.(Braun et al., 2017)). as well as the ability of applications built with them to handle particular dialog structures, such as subdialogs (Larsson, 2017) and question answering behavior (Larsson, 2015). Our focus is on the tools themselves and our goal is to not just look at what they do, but what we need them to do.

2 Natural Language Understanding

The most mature component of dialog tools is natural language understanding (NLU). The field has

Proceedings of NAACL-HLT 2019, pages 106–113
Minneapolis, Minnesota, June 2 - June 7, 2019.

settled on the notion that understanding comprises recognizing the user's intent and extracting the entities in the utterance required to fulfill the intent. While the terms harken back to Barbara Grosz and Candy Sidner's seminal paper "Intentional and Attentional Structure" 1986, the implementation can be more closely tied to work at AT&T, which combined statistical classification used for call routing in applications like HMIHY (How May I Help You) with mainly rule-based extraction of entities (Gupta et al., 2006). Nuance 9 tools allowed both utterance classification ("Statistical Semantic Models") and partial parsing for information extraction ("Robust Parsing Grammars") commercially over a decade ago (Nuance, 2002), though a single utterance could only be processed by one or the other, not both, which significantly limited effectiveness compared to today's systems.

In addition to being fairly mature, NLU performance can also be evaluated quantitatively. We compared three tools, Google's Dialogflow[1] (previously api.ai), IBM Watson Assistant[2], and Microsoft LUIS[3].

We choose two domains to compare. The first is a tech support IT FAQ system which initially included a fairly limited set of questions with answers designed by the customer. Some of these questions had multiple answers based on information such as what kind of computer the user had. The second lets users ask questions about their own financial statements, so the questions are limited by the information available on the account and the answers are user-dependent. Both are currently in trial with customers.

We report results on our regression tests rather than the live application. All of the questions in the test sets were "fully specified" in that there was no missing information that needed to be filled in through dialog.

[1]https://dialogflow.com
[2]https://www.ibm.com/watson/ai-assistant/
[3]https://www.luis.ai

Domains	IT V1	IT V2	Finance
Intents	23	51	14
Entities	30	44	15
Training	1164	14525	426
In domain test	267	450	113
OOD test		150	100

Table 1: Test domain numbers

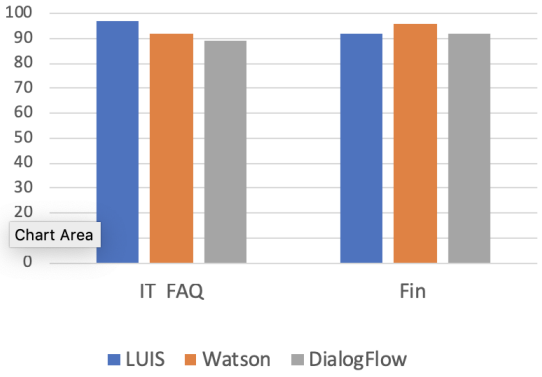

Figure 1: Intent Accuracy

Our first set of tests compared performance between LUIS, Watson and Dialogflow on the two domains. Details of size, training and test are shown in Table 1. We were able to do a direct comparison of intent recognition across the three systems, since we wrote acripts that translated a single source of intents, examples, and entities into the formats required for the tools. Figure 1 shows the performance across the three tools on for IT FAQ V1 and FIN. They were relatively close, with LUIS and Watson each taking first place depending on the domain.

2.1 Intents and Training

In this section, we discuss the effects of increase in scope and increase in training data. Due to time constraints we focused this work on just the IT domain tested on LUIS and Watson. We has more than doubled the scope of the IT application and used both data collection and automatic paraphrasing to increase the amount of training data.

Collecting data when operating in a rapid production mode is challenging. We used Google forms and other tools to collect "paraphrases" from SMEs (subject matter experts) internally and from the customer. One of our first discoveries was that not not everyone knows what a paraphrase is and many of the samples were related questions, for example "How do I disable automatic software updates?" was given as a paraphrase of "How do I get automatic software updates?"

2.1.1 Augmented Training Data

In order to further increase the training data, we used a grammar-based data augmentation technique for generating additional paraphrases for the

Training	Training	LUIS	Watson
V1: Human	1164	63.2%	70.2%
V2: V1+Auto	14525	73.0%	75.5%

Table 2: Intent accuracy comparing training amounts

IT domain. The grammar contained 58 equivalencies, where each equivalency consisted of two or more synonymous words or phrases.

An example of an equivalency is shown here, with alternatives separated by semi-colons:

> **should i**: is it ok to;am i allowed to;do i have permission to;is it bad to;is it against company policy to;is it okay to

In order to generate paraphrases, we create two copies of each sentence, the unaltered input and a processed version of the sentence, each labeled with the intents and entities from the original. The script looks for instances of entries in each equivalence class. If one is found, it randomly replaces that entry with one of the other entries in that class. If no entries of any equivalence class are found in a particular sentence, the paraphrase is identical to the original. After running the script the desired number of times, we take the set of unique sentences in the output as our augmented dataset. Table 2 shows performance of LUIS and Watson on V1 (human created training only) and V2 (human plus automatic paraphrases).

2.1.2 How much is enough?

We also experimented with running the script between one and four times, giving us the potential for up to 16 times the amount of data in our original set. After eliminating duplicates, though, the amount of data is smaller, as shown Table 3 [4]:

We built models in Luis for each of these training sets and test on a common test set of 600 sentences. Results are shown in the third column of the table.

Because the test set is generated from the same methodology as the training set, it lacks some variability which we expect to see in a deployed system. As an attempt to model that variability, we experimented with also passing the test set through the same paraphrase generation module as the training set. Doing so resulted in a new test set with 3005 sentences. Results on that augmented test set are shown in the 4th column of Table 3.

By looking at results on the unaugmented test together with the results on the augmented test, we decided that the set of 7528 training utterances resulted in a desirable model in that it did not degrade the performance on the original test set and resulted more robust performance on the augmented test set.

2.2 Out of Domain and Scope

A significant challenge in a deployed system is recognizing when an utterance is out of domain (OOD), and thus can't be answered. This includes questions:

- Questions from a different domain, e.g. asking the IT FAQ application about the weather,
- Questions in the domain, but not in the scope of the application, e.g. the IT FAQ might be able to help you get a VPN account, but not a Jira account.
- "Adversarial" questions, that is intentially trying to break the system, e.g. "Can you eat VPN?" and "What do you look like".

The choice of the domain and scope in a commercial system is defined largely by the customer but not always clear to the end users, who may think the IT FAQ can help them get any kind of account or the app can answer the same things as Siri.

In the finance domain we used a 100 utterance OOD test set and introduced a new intent for classifying these utterances as "None". The "None" category had 28 training utterances, which was about average for the categories in V1. The second set in Figure 2 combines the in and out of domain test sets and the third set shows results on

No. Iter	No. Sents	Intent Acc.	
		Orig. Test	Aug. Test
0 (orig)	2296	72.8	69.9
1	3086	73.5	73.9
2	4789	74.2	76.1
3	7528	73.0	76.9
4	10719	71.8	76.5

Table 3: Results of data augmentation on the standard test set.

[4] While results within a table are consistent, different sets of results may differ given the evolving nature of commercial systems.

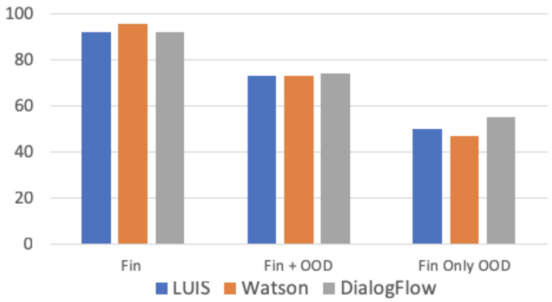

Figure 2: Intent Accuracy In and out of domain

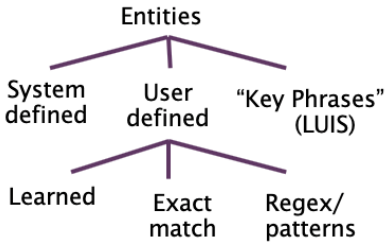

Figure 4: Entity Types

just OOD. It is clear that performance drops significantly in all the tools due to the OOD effect.

Figure 3 shows the correct accept vs. correct-reject curves for each of the tools as we threshold the response based on confidence scores. The Watson score has better confidence scores than the other tools.

While overall we find the tools quite good at classifying in-scope intents, their performance is poor on detecting out of domain and scope, which are admittedly difficult to model with a classifier. However, on the positive side, the tools allowed us to get a system into the hands of users quickly without having to start with a massive data collection effort. As we get more user data to help identify kinds of out of domain and out of scope questions will be asked, we will be able to improve on the base performance.

2.3 Entities

All three tools allow entities to be defined in multiple ways, as shown in Figure 4. System entities are common types, such as time and currency that have been built and trained by the creators of the tools. Regular expressions can be defined by application developers to capture structured strings, like order numbers. The most interesting distinction is between learned and exact match entities. Machine learned entities are able to identify synonyms without having to predefine them. For example, if "passphrase" is used in the same context as "password" the system should be able to recognize it as a synonym.

In Dialogflow, the predominant method of defining entities is exact match, which defines entity types with values and synonyms. Learned entities appear to be fairly limited and used for things like items on a shopping list.

In LUIS, exact match entities are "List" entities, which also has a type, values and synonyms. Machine learned entities, which take context into account, are call "simple". In order to capture type and values, you can use "hierachical" entities. For example, in the IT domain, the "Investment Account" entity has as its children the five different investment accounts. The accounts had fairly complex names which were frequently reduced, for example the "Membership Contribution Pension Fund" could be referred to as the "Membership Fund", "Membership Contribution Fund", "Pension", etc. We implemented these both as list and learned entities, as shown in Table 4. List had better performance, but required listing all the variations. Learned requires annotation, which is also a cost, but it is not the case that every variant has to be accounted for.

Watson has a beta feature that lets you annotate entities in context, including annotating counter

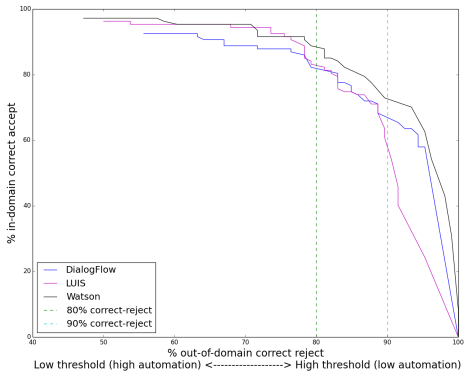

Figure 3: Confidence Scores

	Train	Test	List	Learned
Account	228	61	96.3%	93.8%
Slot	102	26	100%	91.5%

Table 4: Test for Entities and Slots

examples, which may be able to be used to improve performance, though due to bandwidth we were not able to do a comparison.

2.3.1 Slot filling

Slot filling is a core capability in NLU: recognizing not just that an entity occurred, but which parameter or "slot" it is filling. In most cases, which slot is being filled can be determined by the type of the entity, so as long as it is correctly identified, the slot is filled correctly. However in some cases one entity type can fill two different slots, as in transferring funds from Account A to Account B.

As with entity recognition, all three tools provide slot filling functionality we found that more rule based methods of identifying slots was more accurate than learned slot filling (see Table 4).

2.3.2 Recognizing new entities

Anything that does not match an entity is ignored. While this simplifies understanding, it can lead to false positives. For example, the question "What's my balance" in the finance app applies to the entire portfolio if no specific account is mentioned. However, if an account is mentioned but not accurately recognized, the answer will be incorrect.

A notable exception of this is LUIS, which has the unique ability to find entities that haven't been predefined through "key phrases". This additional information can be used in flagging problem interpretations and used in clarification dialogs, as in the example "How do I get a Jira account?", the intent ProcedureGetAccount is triggered and a key phrase "Jira account" is found, which would allow a clarification response such as I think you're asking about how to get an account, but I don't know anything about a "Jira account".

2.4 Language Contraction

An essential part of language understanding is the ability to interpret utterances when they undergo what we might call "language contraction" where some parts are left out or replaced with pronouns or reduced forms. While as we show in Section 3, dialog structure can be used to emulate language contraction interpretation, none of the tools address the phenomenon directly. There are two kinds of constructions that need to be addressed for dialog tools to be able to effectively handle conversational dialog.

Coreference: Where a pronoun or other referring expression is directly referring to some previous entity in the sentence.

- Whats the balance in my retirement account? When does it vest?

Ellipsis: Part of the utterance is left out and can be filled in by some portion of an earlier sentence.

- What are the fees on my retirement account? How about my annuity?

While none of the tools address this directly, both Watson and Dialogflow keep track of values in context variables. For example, in Watson, once you ask about one investment account, the variable `investmentAccount` is set, so the sequence below would be answered correctly:

Whats the value of my portfolio plan?

What are the fees?

How much as it changed over the past quarter?

However, if you asked about an account it didnt understand, such as "What are the fees for my fidelity investments", it would miss that and answer based on the previous setting of the `investmentAccount` variable.

3 Dialog Management

Beyond the level of interpretation of single utterances, the tools differ significantly. In this section we look at the development of the dialog structure and the management and application of context. To illustrate the various capabilities, we use a real estate application, which has more complex interactions. We compare the approaches of Dialogflow and Watson Assistant. The Microsoft offering has three separate tools each with their own interface and would have been a steep learning curve for developers who are not already well-versed in the Microsoft landscape, so we did not pursue it further.

A significant difference between the approaches in the tools is how tightly integrated the dialog capabilities are with the NLP. In Dialogflow, the intents are the basic dialog components and both context and control are defined as part of the intent definition, whereas Watson Assistant provides a separate interface that explicitly allows dialog nodes to be defined independently from intents and entities.

Defining dialog structure independently from the intents can be advantageous. First, the condition of whether a dialog node is triggered can be

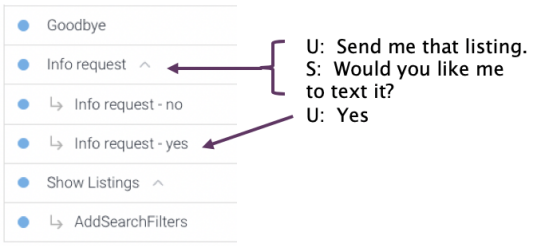

Figure 5: DialogFlow

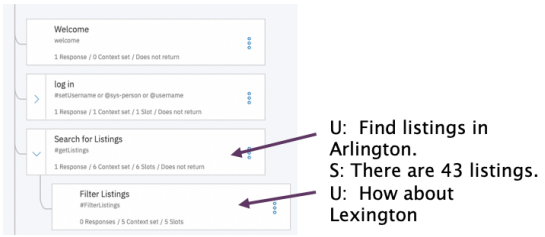

Figure 6: Watson

dependent on variables outside of the interpretation of the utterance, for example checking a context variable to determine whether a user is logged in. Second, an intent can function differently depending on where in the dialog that intent occurs.

The dialog structure allows users‘ answers, which are dependent on what question is asked, to be interpreted correctly, as seen in Figure 5.

Dialog structure can allow for the interpretation of coreference and ellipsis, as shown in Figure 6. Neither tool has any model of coreference or anaphora, however, careful construction of the dialog structure ensure that the utterances are interpreted correctly.

3.1 Frame-based Dialog Management

Frame-based dialog management uses the slot filling in a frame or form to control the dialog. The advantage is that it allows mixed initiative interaction: users can provide the slot values in any order and "package" them into any grouping. For example, the `CalculateMortgagePayment` intent requires an amount, term, and interest rate. The user might start by saying "What is the monthly payment for $500,000 mortgage over 30 years at 4% interest" or just "How much would the monthly payment be for $500,000". The dialog manager fills slots as the information is provided and then asks the user specifically for information that it doesn't have yet. In this example, the dialog manager might ask for a rate, but the user might choose to respond: "I am looking for a 30-year mortgage".

In contrast, a traditional procedural dialog manager covering all the possible response sequences would require complicated software development and maintenance. While only very narrow domains can be fully controlled with a frame-based mechanism, it is a very useful control structure for a range of intents which have multiple required slots to fill.

In Dialogflow, frames are defined as part of the definition of the intents. Entities are explicitly associated with the intent and any subset can be marked as "required" and the developer provides a prompt which will be asked if the entity is not filled. One downside is that the frame control can't be interrupted. For example, if the system says "What zip code would you like to search?" the user can't ask "What's the zip code for Arlington?" before answering. There is also no capability for prompting for multiple required parameters at the same time. The questions are simply strings rather than being composed dynamically, so context can't be taken into account.

Watson Assistant defines frames at the node level. The functionality is similar, allowing the app developer to add prompts for required entities. It also allows for a single prompt to be defined if none of the slots have been filled. In addition, there are "Handlers" that can interrupt the process if the user doesn't provide the required information.

3.2 Context Management

Context plays an important role in both interpretation and control of the dialog, for example the dialog manager needs to:

- Keep track of variable values that are used later, such as the user's name.
- Record that a particular action was completed, such as logging in, so that it can be checked later as a condition for another action.
- Collect information over multiple turns, such as search criteria to narrow results.

Intents in Dialogflow have input and output context variables to gate actions as well as to store the values of entities. We found some difficulties in keeping values in the context over multiple turns.

Watson assistant used the dialog structure to manage the context. While this made it clearer, there was no easy way to control what is kept in the context. However, there is a view of context variables and values in the tool interface which is useful in the design process and debugging.

3.3 Challenges in Dialog

While Watson Assistant's use of a dialog tree provides greater functionality, it still falls short when attempting to build a fully mixed initiative system. "Digressions" are allowed in Watson, but need to be defined in advance, so in the zip code example above, the application developer would need to put an explicit digression to a node for `FindZipcode` which would then return to the original dialog node. While this one seems fairly obvious, it quickly becomes difficult to predict any place the user could digress from a conversation.

4 Response Generation

Text generation is a hard problem and has gotten neither the attention nor funding that understanding and other subfields of natural language have gotten over the years. It is not surprising that it is the least developed part of the tools. Within the tools, the responses are created by the app developers, either just as word strings, mostly for prompts, or templates with variables that correspond to entities.

Both tools allow multiple responses to be created that will be chosen from randomly, creating some variability, but only Watson assistant allows the responses to be contextually dependent, based on the values of context variables.

The problem with these approaches is not just the lack of variability, but also since responses are merely strings with some parameters, there is no representation of the constituent structure or the meaning of the response. In the following exchange, "that" refers to the VP constituent corresponding to the procedure of getting an account. However, this constituent is not available for coreference.

S: In order to do save listings, you need to get an account.

U: How do I do that?

4.1 Error Recovery and Clarification

Successful dialog systems require the ability to know when the system did not understand or is not able to answer. We looked at out of domain/scope identification in Section 2.2, but the challenge is in extracting sufficient information to make an intelligent response (Stoyanchev et al., 2016). For example a "reprise question", which is a clarification that repeats a portion of the original utterance, requires that some portion of the utterance to be understood, as in the following, which would need a `ProcedureGetAccount` intent to be recognized in order to answer with one of the two reprise clarifications in the example:

U: How do I get an account on Jira?

S: How do you get what?

U: How do you get what kind of account?

Another important element in good error recovery dialog is not simply repeating the same error message over and over. Best practice in developing speech IVRs is to keep track of both the type of error (e.g. no input vs. low confidence) and the number of tries the user has made in order to vary the error message (e.g. first time, "say again", second, "try rephrasing", and third, type or just move on). While we didn't find this capability in Dialogflow, Watson allows the choice of which response to give to be based on the value of a context variable, which can be set as a counter.

Dialogflow's "Default Fallback Intent" and Watson's "Anything else" trigger when no other intent or dialog node triggers, however, without context there is no ability to tailor the response. Again, Watson's use of an explicit dialog tree allows the developer to have multiple "fallbacks". Within each subdialog, the final node can be set to fire if none of the other conditions hold.

4.2 Conclusion

We provided results and examples across a small number of domains to illustrate the challenges in developing a dialog system that supports building easy to use, natural conversational applications.

The rank-ordered accuracies of the tools varied across domains. While some differences exist among the tools in terms of how entities are handled, more significant differences among them lie in how tightly coupled the NLP and dialog components are.

At our company, we focus on inventing and implementing additional modules with enhanced complimentary features and functions to improve the utility of the three dialog systems that we presented in this paper.

References

Daniel Braun, Adrian Hernandez-Mendez, Florian Matthes, and Manfred Langen. 2017. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 174–185.

Barbara J Grosz and Candace L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

Narendra Gupta, Gokhan Tur, Dilek Hakkani-Tur, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Gilbert. 2006. The at&t spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):213–222.

Staffan Larsson. 2015. The state of the art in dealing with user answers. In *Proceedings of the 19th Workshop on the Semantics and Pragmatics of Dialogue. SEMDIAL*, pages 190–191.

Staffan Larsson. 2017. User-initiated sub-dialogues in state-of-the-art dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 17–22.

Nuance. 2002. *Advanced Developers Guide*.

Svetlana Stoyanchev, Pierre Lison, and Srinivas Bangalore. 2016. Rapid prototyping of form-driven dialogue systems using an open-source framework. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 216–219.