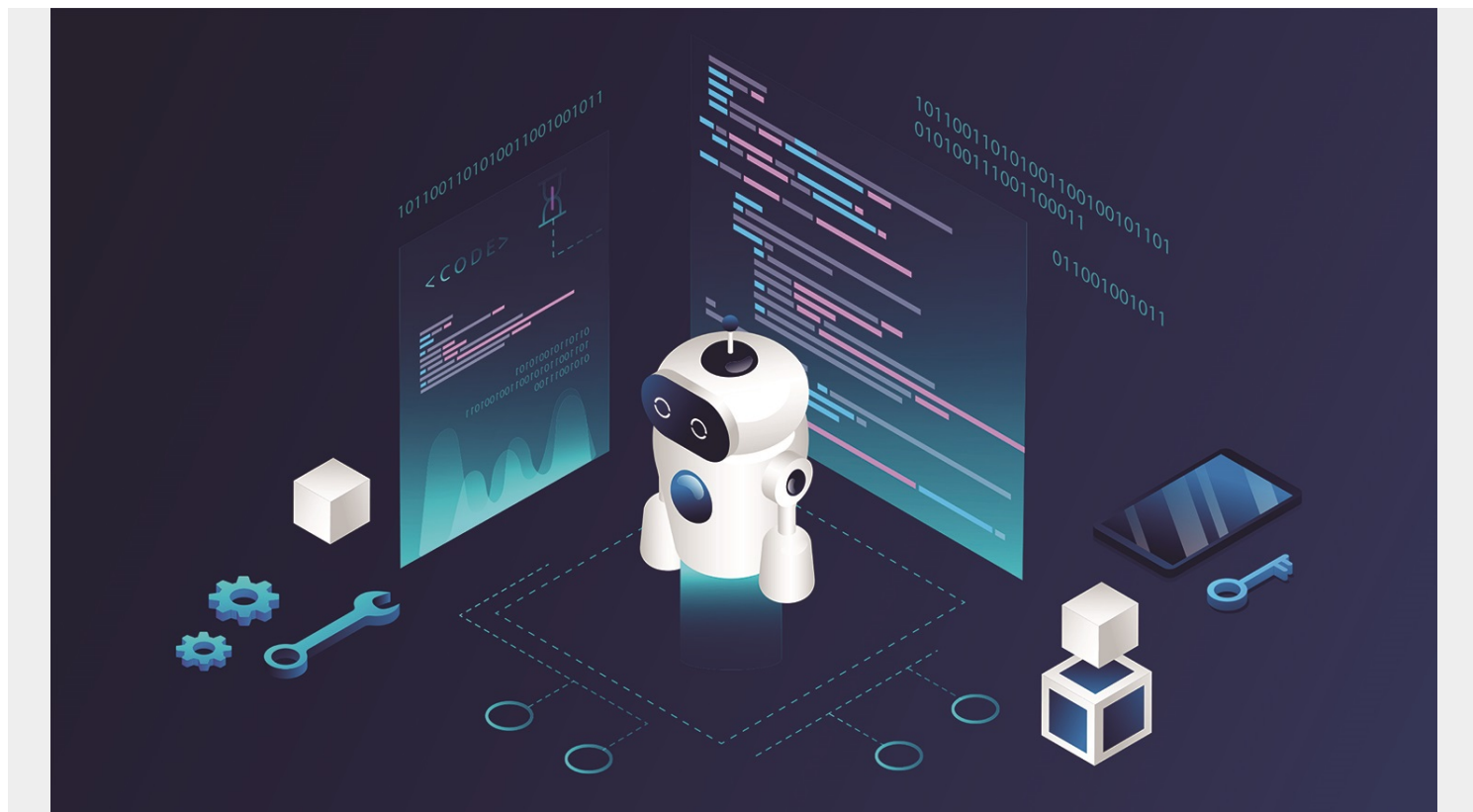


AN ALTERNATE APPROACH TO TRAINING VIRTUAL AGENTS



Chatbots have been around for a long time, since WeChat pioneered the idea over a decade ago. Within the past five years, however, chatbots have seen an exponential increase in use.

Many factors lined up for this to happen. First, consumers became more comfortable communicating on messaging channels, using apps such as Messenger, WhatsApp, Telegram, etc. Next came large platforms such as Facebook opening their messaging API for businesses to build chatbots and finally, an improvement in AI algorithms and affordable compute power drove increased use. Voice-based assistants such as Alexa and Google Home made it an absolute must for businesses to use the same conversational communication channels that customers would use.

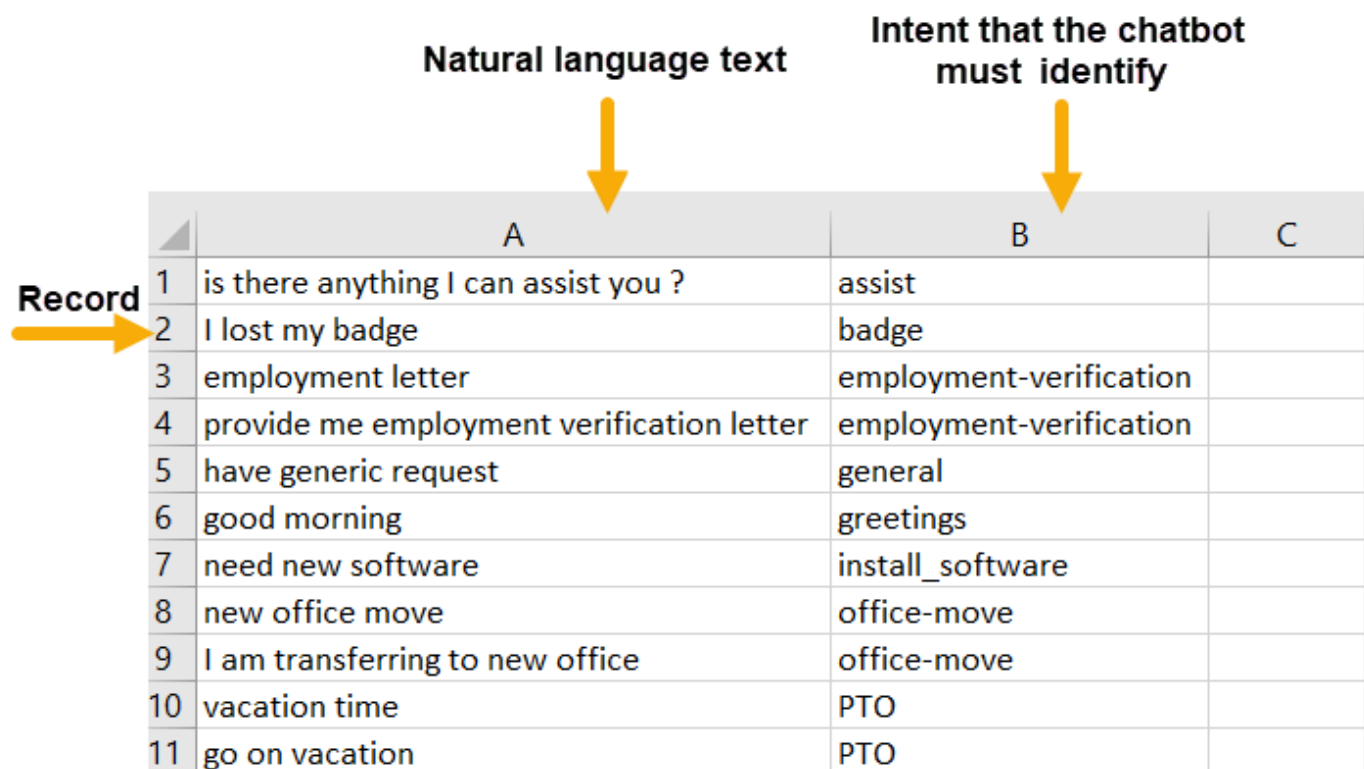
At the peak of the chatbot hype, there were many companies offering to build chatbots, and most messaging platforms had similar opportunities on their platforms. A fair bit of consolidation has happened since then, but the chatbot market is still highly fragmented, with both vertically focused organizations as well as big cloud players offering their general-purpose conversational platforms.

I was fortunate to have been in the driver's seat of building conversational AI platforms since the early stages of the chatbot hype cycle and want to share some of the lessons I've learned.

How most platforms are trained today

If you take any modern conversational AI platform, there is a common pattern in how they are trained. Every single one of those platforms requires you to provide annotated training data to

classify the intent of anything a user may input. This data is just a collection of sentences or phrases that users might say and what intent it belongs to.



	Natural language text		Intent that the chatbot must identify	
	A	B	C	
Record	1	is there anything I can assist you ?	assist	
	2	I lost my badge	badge	
	3	employment letter	employment-verification	
	4	provide me employment verification letter	employment-verification	
	5	have generic request	general	
	6	good morning	greetings	
	7	need new software	install_software	
	8	new office move	office-move	
	9	I am transferring to new office	office-move	
	10	vacation time	PTO	
	11	go on vacation	PTO	

This is a great approach to start building a chatbot. Today, any small business can get a chatbot up and running in less than a day, set up by a novice programmer.

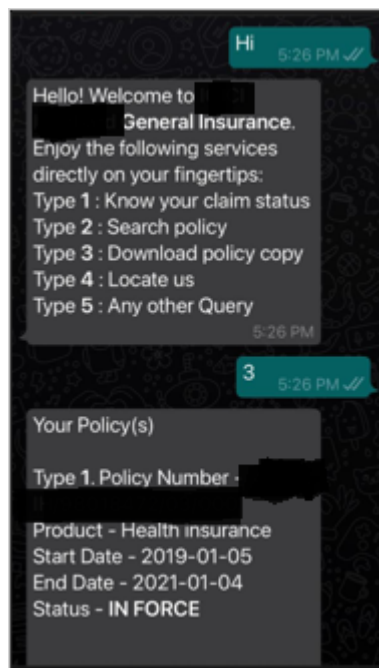
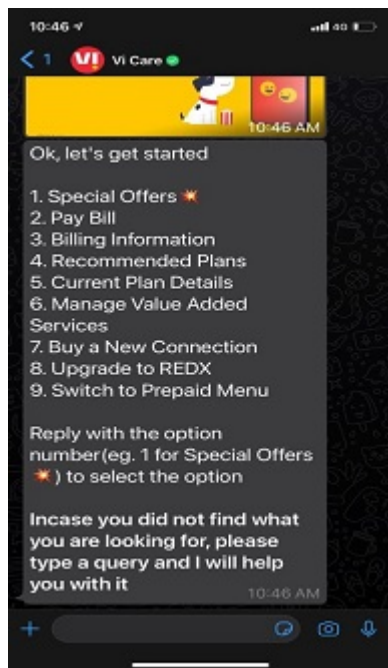
Challenges arise when you attempt to build virtual agents, such as what BMC does in the ITSM domain, or what I used to do in the banking domain at my previous role.

Training a chatbot with annotated intent data for classification will become exponentially complex as the number of intents and amount of training data increases. Very soon, it becomes nearly impossible to continually maintain the chatbot or further improve it. Over the past five years, I have had the growing feeling that there had to be a better way.

Pretrained models such as BERT (Bidirectional Encoder Representations from Transformers) have made the chatbot training process much more accurate at an enterprise scale, with less data. But the challenge of manageability remains.

Failures of the current training approach

I believe we may have taken the wrong approach to the classification problem. The method of using annotated training data with intents helped make the process of building chatbots easier, especially chatbots pulling from simple FAQ data. But take one look at any chatbot today that offers services beyond FAQs, such as transactions or inquires, and you will realise that this approach has failed in most implementations. Many chatbots have ended up no better than IVR systems or keyword SMS/USSD based services, which is pretty embarrassing for the amount of progress we have made in the field of NLU.



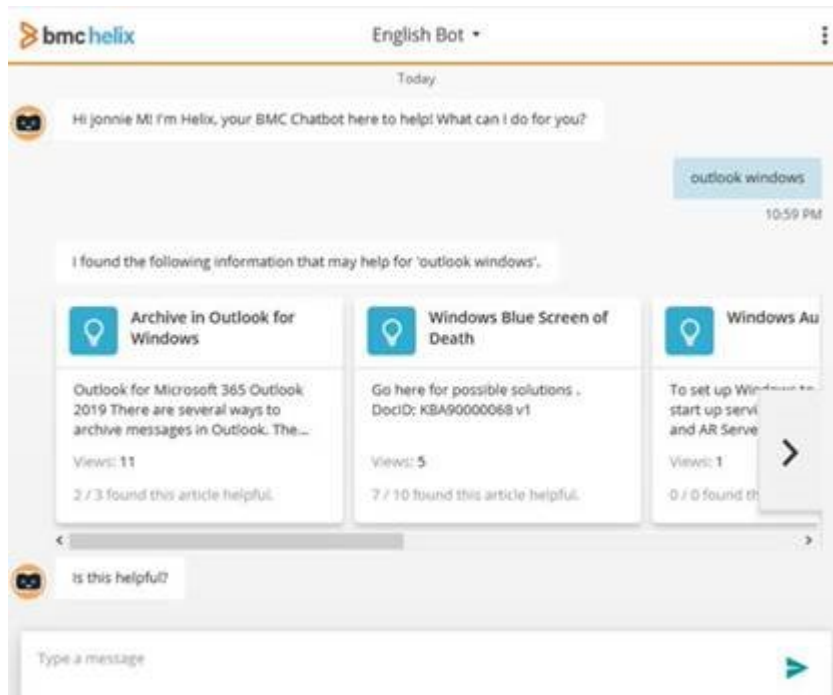
Let's take some scenarios that users might ask for from a virtual agent:

- "When did I change my password?"
- "When do I have to change my password?"
- "Is my account locked?"
- "Unlock my account"

Now, these user inputs scenarios alone are not that complex to build. But if you want a chatbot that can give precise responses to such scenarios, you can easily end up having thousands of intents. Training and maintenance at that scale is a challenge. Integrating to the data sources to pull relevant responses to each of these thousands of scenarios is an impossible task.

How did this happen?

Almost all current data-input approaches are fine-tuned for ease of development and training. They keep the learning curve as easy as possible for the chatbot trainer. Many platforms have taken the approach of using unstructured data sources as knowledge bases to train chatbots, such as documentation. Most of these fail to provide a user experience or accuracy that is even as good as a search engine. They also add to the manageability crisis when bolted on.



Here is something that always bothered me:

if you hire customer support executives for your organisation, how would you teach them?

Would you give them a list of annotated training data and intents? No.

Would you give them a bunch of FAQs with their answers? Most likely, yes.

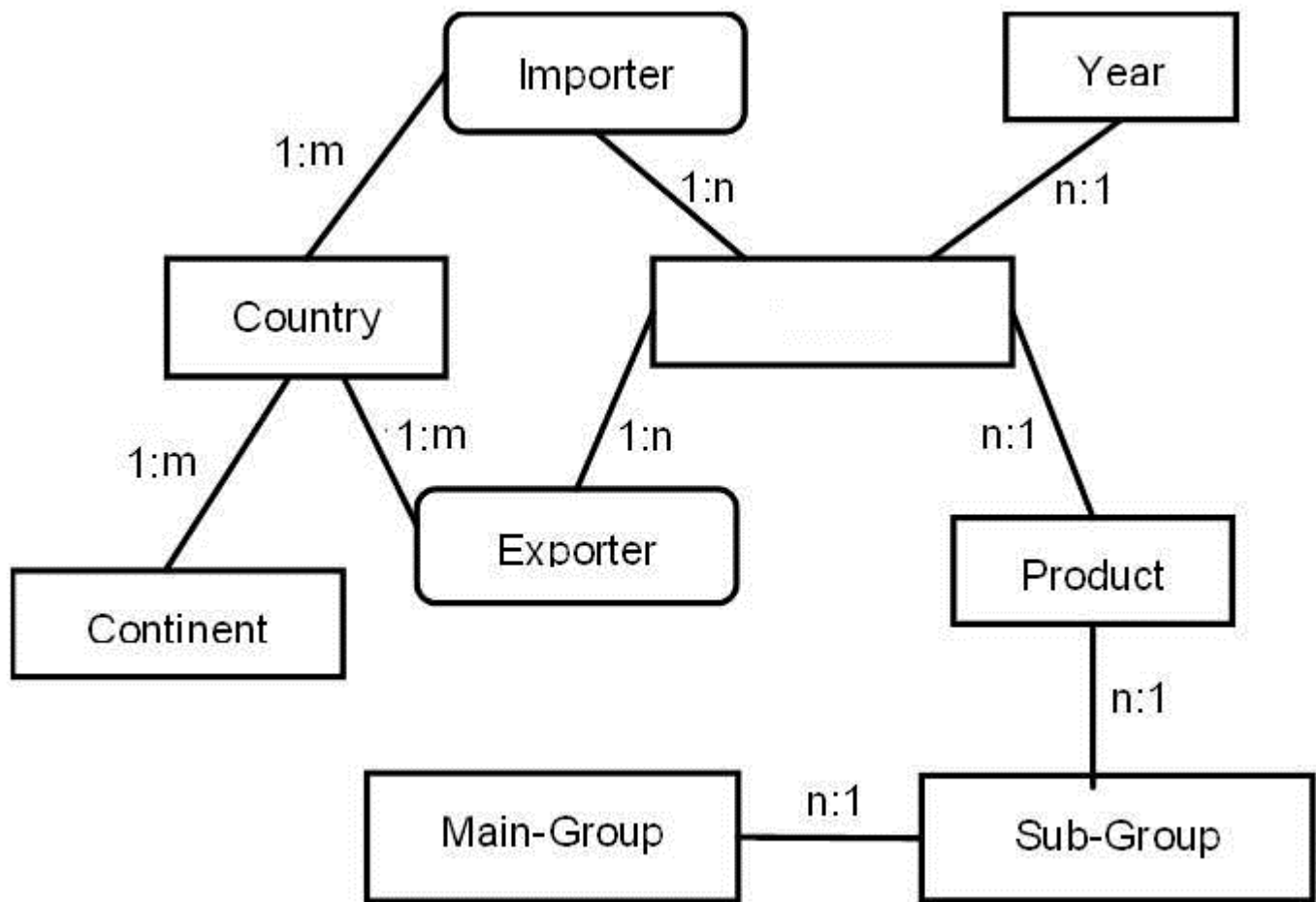
But, first and foremost, you would teach them about the domain, your business, your systems, and where they can find information. Why should teaching machines be any different?

Without your chatbot having an intimate understanding of the domain, organisation, and the systems of record, building a chatbot becomes an exercise in brute force and trying to compensate for the training issues with rules that end up a tangled mess.

A better approach

A much better approach to building such chatbots would be to breakdown learning into a few steps, just like how we would teach humans. Some steps are as follows:

1. Teach the chatbot the language understanding (English, Spanish, etc.). You can use pretrained models of this.
2. Teach the chatbot your domain. This can be done by providing a structured graph of all elements of your domain.
3. Teach the chatbot where it can access the data "enterprise specific or user specific".
4. Teach the chatbot all the frequently asked questions, chitchat, etc.
5. Add domain specific lingo that is not common English. "I want to place a bet against Tesla" which means "Short sell Tesla".



When you do this, the chatbot can understand what the user says in context of the domain, map it to the right data source, see clarifications, and respond to the user. Even when it learns from unstructured sources, the chatbot will be able to better contextualise what it is reading based on its understanding of the domain.

Without getting into too much detail, let me give you a simple example of a technical approach. Let's say you have a database where all user information and history is stored. As long as the chatbot can convert the user's query into an SQL query, it can fetch any data and respond, or even update the data in the database. Technology like this can bridge unstructured free form requests from users into a more structured format for more accurate and precise responses.

In banking, that means a user can ask very unusual queries such as "when did I open my last Term Deposit account?" which will get translated into "select max(opendate) from term_deposits where userid=876 and status=open".

In 2016, Viv showed an [inspiring demonstration](#) of this kind of complexity. It was eventually bought by Samsung, and we haven't seen anything like that available for enterprises to build chatbots.

Conclusion

At BMC, we believe self-service chatbots are the future. For them to be adopted within the enterprise, they should be truly intelligent, and should guarantee ease of building and maintenance of a virtual agent.

It's time we rethink our approaches for building virtual agents. Let's teach chatbots so they actually understand the domain and not just the language.

One benefit of such approaches is that it will reduce false positives in chatbots. Detecting and managing false positives in chatbots that have tens of millions of users is nearly impossible otherwise. We will talk about detecting and managing false positives in my next blog post.

Until then, ciao.