

POINT OF VIEW

NLP usability and performance improvement for Conversational AI

AUTHORS

Rohit Vishwakarma



Abstract

We live in an era of constant digital transformation where more and more brands are using new technologies to reach out to consumers. As much as the need to make sure that services are available to consumers 24/7, the need for automated and quick service-provisioning has become an important factor to be considered. One of the modern approaches to this is **conversational AI**. **conversational AI (cAI)** is an advanced way of offering a conversational experience through digital personal assistants that mimic real-life conversations with people. In other words, services are being offered using Conversational AI chatbots or virtual assistants. Examples of such virtual assistants are 'Amazon's Alexa, Microsoft Cortana, and 'Apple's Siri. While such solutions are great in providing instant service to the end-user, a very common question arises – how do these bots understand human language so well, considering dialects, accents, and other things? Also, a virtual assistant built on a smaller level generally fails to deliver the user experience that these big-brand products deliver with ease.

Generally, a common frustration point when people use chatbots is that the bot 'doesn't always understand what the end-user wants to convey. Then, what is the mantra that these big players in the market are following so well? The answer to all

these questions is **Natural Language Processing** or **NLP**. NLP is a subfield of Artificial Intelligence that focuses on helping computers understand human language in a much-improved manner. The idea of this Point of View is to cover the concept of NLP and how NLP for Conversational AI can be used to improve customer experience and thereby improve overall product engagement.

Now, even before talking about NLP and NLP-based AI models, 'it's worth answering the question...

'Why does having chatbots/virtual assistants increase customer engagement and provide a better user experience?'

So, 'let's have a look at them.

01

Conversational AI benefits

Quick response

An important part of user engagement is the speed of response. It is quite common that a customer query could go to a service engineer, and he/she might take a couple of hours or days, depending on the SLA, to resolve the problem. On the other hand, an AI chatbot feature in the product portal is like a quick shortcut to most common user queries, and the bot usually answers within seconds, thereby reducing customer waiting time.

Human handoff lead

Human handoff is the process where the chat between the conversational chatbot and the end-user gets transferred to an actual human agent. There are times when the conversational chatbot fails to understand the user query and tries to ask related questions to get some understanding. However, even then, if the end-user wants to talk to an actual human agent because of the 'bot's lack of understanding things, then the agent already has a good idea about what the user wants to know from the lead provided by the bot during the human handoff procedure.

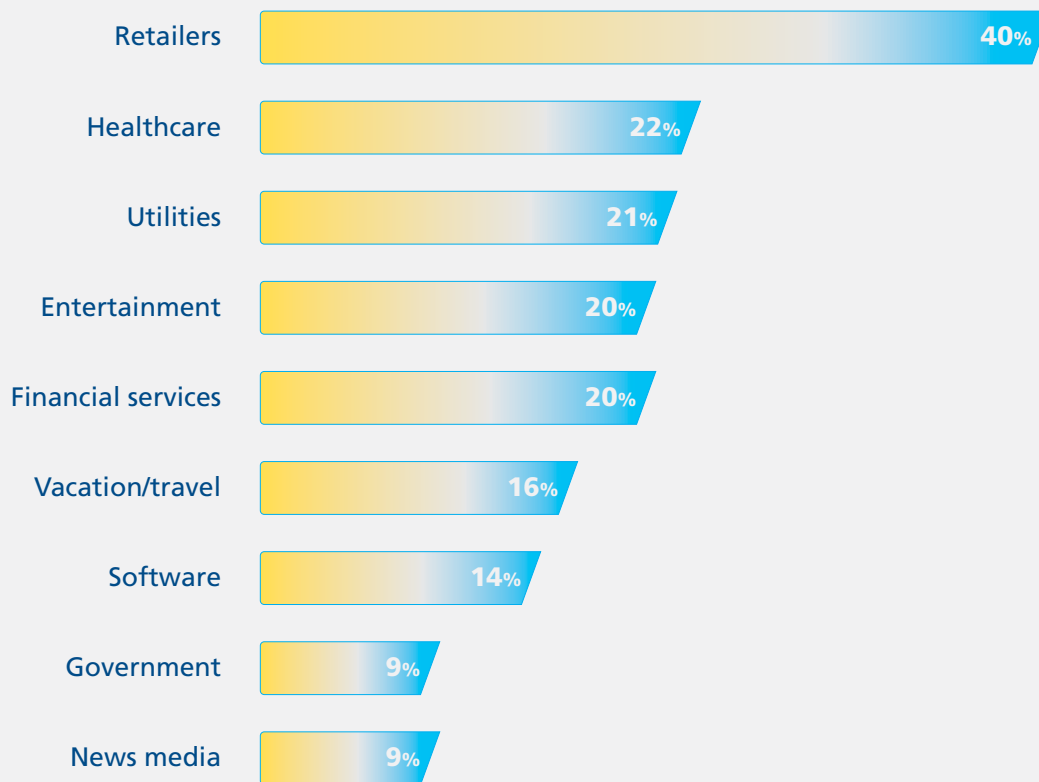
A relief to support agents

Generally, customer queries come as tickets to organizations. As a result, the more the number of customer queries, the more is the number of tickets formed and, ultimately, more responsibilities on the IT support team. With smart Conversational AI chatbots, queries get resolved almost immediately. This leads to lesser tickets being formed and, thereby much relief to the support agents.

Act as Digital Personal Assistants

It is quite often that user engagement improves greatly with personalization. Bots providing personalized services are the go-to-market trends now where a bot almost knows what a particular user usually queries based on the learning process that it carries out with more and more chatting. As a result, these virtual assistants are quite often also termed 'digital personal assistants' and have improved user experience tremendously. To put into context how much influence across sectors in the United States AI conversational chatbots have had in the recent past already, Statista, a leading German company specializing in market and consumer data, published the following graphical information back in 2019.

Share of consumers who have used chatbots to engage with companies in the united states as of 2019, by industry



Source: Landbot

After knowing the benefits of conversational AI and chatbots, let's dive a bit deeper into technicalities and understand the types of chatbots that can be developed and the role of NLP in improving their performances by a great deal.

02 Types of Chatbots

In the simplest classification terms, there are three types of chatbots – simple, smart and hybrid:



Simple

A simple chatbot is a rule-based chatbot that assumes a certain pattern of conversation to take place. It has a specific task-based approach wherein it provides a certain set of predetermined options for the end-user to query from. It also assumes the flow of conversation, and it does not try to learn from previous conversations or anything else. For example, a chatbot built for ordering a birthday cake will be like an operator asking you over the phone about the order. Typically, the person taking the order will ask you the size, flavor, and the name to be written on the cake. Just like that, the chatbot will have rules designed to ask this set of questions, and once the answers are given, the order gets placed successfully.

Here, the bot won't understand any slight deviation from the designed conversational flow', which will lead to a terrible user experience. So, such rule-

based bots generally have simple buttons as query options to choose from for the end-user.



Smart

A smart chatbot is an AI-driven bot where AI transformer-model-based approaches are taken during the design process. Several virtual assistants in the market, such as Siri, Cortana and Alexa are based on this approach. As we all know, these assistants are very smart. They not only understand the human language to a high degree, but they also almost pick up sentiments and dialects and constantly learn from historical data. 'They've grown into popularity because of the customer engagement and the overall user experience they provide.



Hybrid

A Hybrid chatbots are simply the amalgamation of the simple and smart types. In real business case scenarios, these are the most seen and sensible ones to be deployed. They have some rule-based tasks based on their "simpler' side, and they can also understand user intents and context because of their

“smarter’ side. A simple example could be an ITSM ticket assistant bot which provides specific options to query from to the user but also understands if the user has asked something out of the flow. Hybrid chatbots are simply the amalgamation of the simple and smart types. In real business case scenarios, these are the most seen and sensible ones to be deployed. They have some rule-based tasks based on their “simpler’ side, and they can also understand user intents and context because of their “smarter’ side. A simple example could be an ITSM ticket assistant bot which provides specific options to query from to the user but also understands if the user has asked something out of the flow. Say, ‘let’s say the bot provides options to check the tickets assigned, check ticket status, and show SLA-breached tickets. A user then clicks on the button, which instructs the bot to check the status of a particular ticket. The

bot then asks for the ticket ID to the user. At such a time, instead of providing the ticket ID, let’s say the user writes “goodbye’ in the chat box. Now, this is a deviation from the expected conversational flow. However, the bot still understands that the user is saying goodbye and will reply accordingly.

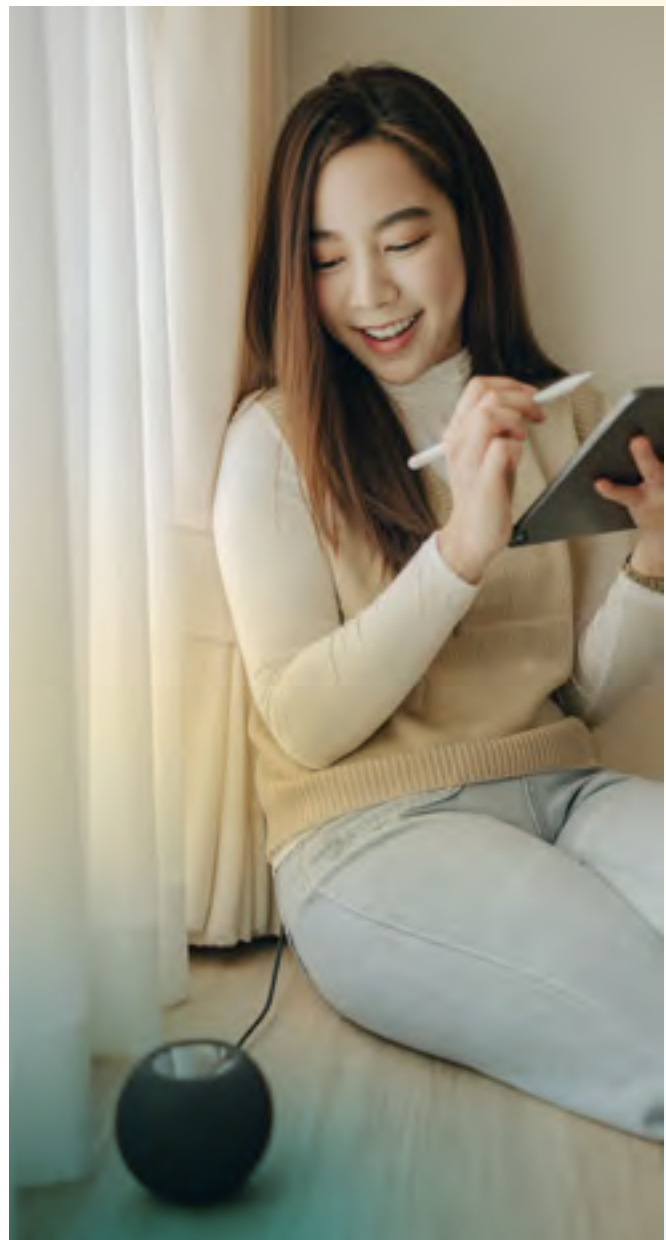
Now, it is pretty clear from the differences above that rule-based bots alone can only function to serve a certain pattern of conversation. Smart and hybrid approaches are the ones that most organizations implement to engage more and more people. The only reason for this is AI, particularly NLP or Natural Language Processing, which is a form of AI that gives machines the ability to not just read, but to understand and interpret human language. Let’s take a closer look at how NLP works actually and improves the performance of chatbots.



03 Natural Language Processing

As mentioned earlier, NLP is for understanding and interpreting human language. With NLP, machines can make sense of written or spoken text and perform speech recognition, sentiment analysis, and automatic text summarization.

Now, as mentioned, NLP involves various tasks, but it totally depends on the requirements we need during our virtual assistant's development process. For instance, advanced virtual assistants like Alexa, Siri, etc., require speech recognition where the bot needs to perform speech-to-text conversion. In other cases, speech recognition is not required; hence, that task under NLP can be neglected for that particular purpose. Similarly, not every chatbot needs translation, so 'we'll just look at a few important model approaches used almost everywhere.



04 How does NLP help in understanding the human language?

Every NLP task requires one or more existing or developed ML models to simplify the process.

Before moving forward, it is good to understand what some of the most critical tasks are to be performed by a machine before even proceeding to understand the human language. Once a human enters his or her intent as a sentence, the first thing needed to be done here is to separate each word and put them in a bag of words, symbols, etc. This is called tokenization. Then comes lemmatization, which is extracting the base form of a comment, so dancing, danced, etc, have a common root word ‘dance’. Then once these things are done, stepping into the following process of understanding the human language, entity recognition plays a vital role. Human intents might or might not consist of an entity. So, for example, the sentences – ‘I’m looking for a hospital in Mumbai’, ‘Can you look for a hospital in Bangalore’ have a ‘location’ entity i.e., Mumbai and Bangalore respectively. Named Entity Recognition is a process under NLP which handles this. Alongside this, we have Part of Speech Tagging, which classifies the tokens extracted from the intent into various parts of speech such as nouns, verbs, and adjectives.

Today, one of the most popular frameworks for developing virtual assistants is Rasa. It uses Python programming language and has a rich feature of developing one’s pipeline-based approach for understanding human inputs and managing conversations. Rasa has two major parts – Rasa NLU (Natural Language Understanding) for understanding human intents and entities and Rasa Core for dialogue management. The NLP tasks discussed above are taken care of by the pipeline defined for Rasa NLU. Rasa has a configuration file where already developed models can be pipelined and used as per the requirement to produce a final trained NLU model. Let’s have a look at an example NLU configuration.

```

1 # Configuration for Rasa NLU.
2 # https://rasa.com/docs/rasa/configuration
3 # https://rasa.com/docs/rasa/configuration
4 # https://rasa.com/docs/rasa/configuration
5 # https://rasa.com/docs/rasa/configuration
6 # https://rasa.com/docs/rasa/configuration
7 # https://rasa.com/docs/rasa/configuration
8 # https://rasa.com/docs/rasa/configuration
9 # https://rasa.com/docs/rasa/configuration
10 # https://rasa.com/docs/rasa/configuration
11 # https://rasa.com/docs/rasa/configuration
12 # https://rasa.com/docs/rasa/configuration
13 # https://rasa.com/docs/rasa/configuration
14 # https://rasa.com/docs/rasa/configuration
15 # https://rasa.com/docs/rasa/configuration
16 # https://rasa.com/docs/rasa/configuration
17 # https://rasa.com/docs/rasa/configuration
18 # https://rasa.com/docs/rasa/configuration
19 # https://rasa.com/docs/rasa/configuration
20 # https://rasa.com/docs/rasa/configuration
21 # https://rasa.com/docs/rasa/configuration
22 # https://rasa.com/docs/rasa/configuration
23 # https://rasa.com/docs/rasa/configuration
24 # https://rasa.com/docs/rasa/configuration
25 # https://rasa.com/docs/rasa/configuration
26 # https://rasa.com/docs/rasa/configuration
27 # https://rasa.com/docs/rasa/configuration
28 # https://rasa.com/docs/rasa/configuration
29 # https://rasa.com/docs/rasa/configuration
30 # https://rasa.com/docs/rasa/configuration
31 # https://rasa.com/docs/rasa/configuration
32 # https://rasa.com/docs/rasa/configuration
33 # https://rasa.com/docs/rasa/configuration
34 # https://rasa.com/docs/rasa/configuration
35 # https://rasa.com/docs/rasa/configuration
36 # https://rasa.com/docs/rasa/configuration
37 # https://rasa.com/docs/rasa/configuration
38 # https://rasa.com/docs/rasa/configuration
39 # https://rasa.com/docs/rasa/configuration
40 # https://rasa.com/docs/rasa/configuration
41 # https://rasa.com/docs/rasa/configuration
42 # https://rasa.com/docs/rasa/configuration
43 # https://rasa.com/docs/rasa/configuration
44 # https://rasa.com/docs/rasa/configuration
45 # https://rasa.com/docs/rasa/configuration
46 # https://rasa.com/docs/rasa/configuration
47 # https://rasa.com/docs/rasa/configuration
48 # https://rasa.com/docs/rasa/configuration
49 # https://rasa.com/docs/rasa/configuration
50 # https://rasa.com/docs/rasa/configuration
51 # https://rasa.com/docs/rasa/configuration
52 # https://rasa.com/docs/rasa/configuration
53 # https://rasa.com/docs/rasa/configuration
54 # https://rasa.com/docs/rasa/configuration
55 # https://rasa.com/docs/rasa/configuration
56 # https://rasa.com/docs/rasa/configuration
57 # https://rasa.com/docs/rasa/configuration
58 # https://rasa.com/docs/rasa/configuration
59 # https://rasa.com/docs/rasa/configuration
60 # https://rasa.com/docs/rasa/configuration
61 # https://rasa.com/docs/rasa/configuration
62 # https://rasa.com/docs/rasa/configuration
63 # https://rasa.com/docs/rasa/configuration
64 # https://rasa.com/docs/rasa/configuration
65 # https://rasa.com/docs/rasa/configuration
66 # https://rasa.com/docs/rasa/configuration
67 # https://rasa.com/docs/rasa/configuration
68 # https://rasa.com/docs/rasa/configuration
69 # https://rasa.com/docs/rasa/configuration
70 # https://rasa.com/docs/rasa/configuration
71 # https://rasa.com/docs/rasa/configuration
72 # https://rasa.com/docs/rasa/configuration
73 # https://rasa.com/docs/rasa/configuration
74 # https://rasa.com/docs/rasa/configuration
75 # https://rasa.com/docs/rasa/configuration
76 # https://rasa.com/docs/rasa/configuration
77 # https://rasa.com/docs/rasa/configuration
78 # https://rasa.com/docs/rasa/configuration
79 # https://rasa.com/docs/rasa/configuration
80 # https://rasa.com/docs/rasa/configuration
81 # https://rasa.com/docs/rasa/configuration
82 # https://rasa.com/docs/rasa/configuration
83 # https://rasa.com/docs/rasa/configuration
84 # https://rasa.com/docs/rasa/configuration
85 # https://rasa.com/docs/rasa/configuration
86 # https://rasa.com/docs/rasa/configuration
87 # https://rasa.com/docs/rasa/configuration
88 # https://rasa.com/docs/rasa/configuration
89 # https://rasa.com/docs/rasa/configuration
90 # https://rasa.com/docs/rasa/configuration
91 # https://rasa.com/docs/rasa/configuration
92 # https://rasa.com/docs/rasa/configuration
93 # https://rasa.com/docs/rasa/configuration
94 # https://rasa.com/docs/rasa/configuration
95 # https://rasa.com/docs/rasa/configuration
96 # https://rasa.com/docs/rasa/configuration
97 # https://rasa.com/docs/rasa/configuration
98 # https://rasa.com/docs/rasa/configuration
99 # https://rasa.com/docs/rasa/configuration
100 # https://rasa.com/docs/rasa/configuration

```


Let's go line-by-line and understand each model used here for the NLU pipelining:

White space tokenizer – As discussed earlier, tokenization divides the given sentence into tokens. A token is a single word, character, symbol, etc. extracted from the entire sentence. So, a white space tokenizer is a tokenizer that splits on and discards only whitespace characters.

Regex featurizer – Regex patterns are used widely for pattern matching. For example, a user input might contain an area zip code; typically, Indian zip codes are six-digit numbers, such as 500067. Regex patterns are defined during bot development in the training data to extract such entities from the sentence. During training, the Regex featurizer creates a list of regular expressions defined in the training data format. For each regex, a feature will be set to mark whether this expression was found in the user message. All features will later be fed into an intent classifier/entity extractor to simplify classification.

Lexical syntactic featurizer – This featurizer adds features to tokens. These features are like checking upper/lower cases of tokens, checking if the token has a title case, checking if token contains digits, etc.

Count vectors featurizer - Creates bag-of-words representation of user messages, intents, and responses. It groups the tokens as per a particular pattern and keeps a list of them. For example, if there are certain tokens that consist only of digits (1234, 4532532, etc.) and no characters (be21, 2ree, etc.) can be featured in one group.

DIET Classifier – Dual Intent Entity Transformer Classifier or DIET Classifier is used to classify user intents and for entity extraction. The DIET Classifier takes care of the classification of intents based on confidence values, so the intent classified with the highest confidence value will be selected as the intent, and corresponding bot actions would be taken.

Entity synonym mapper – Sometimes, people might type two different things for essentially defining the same thing. A simple example of this are the two following sentences – 'I stay in New York City', 'My friend just shifted from Texas to NYC.' Here, most humans would know or recognize that NYC denotes New York City in this context. But bots don't have such a high level of understanding. To eliminate this problem, we can define entity synonym mapper in our pipeline and a list of words we think could be synonyms in our training data. Then, such complications would be taken care of by the Rasa framework.

Response selector – It gives a key-value pair of responses based on the intent. This key value nested JSON not only gives the response but also provides the ranking based on confidence values. The higher the confidence value, the better the ranking, and accordingly, the response is selected.

Fallback classifier – Sometimes, there occurs a scenario where the chatbot might be queried with some random information that it might've never seen before or is unable to relate the intent with a certain degree of confidence value. For instance, if the threshold value defined is 0.7 in the configuration and a user asks a ticket assistant bot - 'Who is the President of Kenya?' Then there's a good enough chance that the confidence with which a particular intent might've been classified would be below 0.7. In such cases, we don't want the bot to give out a random response corresponding to the random low-confidence intent it had selected. So, a better idea would be to place a default response here for which the fallback classifier is defined. In the training data, a default response to such low confidence intents can be defined, for example, the response could be like - 'Unfortunately, I'm not able to understand, please rephrase.' This will be the default response which the fallback classifier will classify.

In the case shown above, the featurizers and classifier models explained together form a pipeline to classify intents and extract entities and other valuable data. Evidently, this approach is much more complex than a traditional rule-based approach. Still, on the other hand it gives a very good outcome in terms of understanding the human language, which the latter can hardly ever achieve. Conversational artificial intelligence (AI) with NLP can be developed and improved with many other model pipelines, including translation libraries like spacy and speech-to-text recognition alongside the ones discussed above. Apart from this, sentimental analysis can also be done because it is often observed that we humans might use a word in a different tone which might change the entire meaning of the sentence. So, the developed assistant also needs to understand human expressions such as sarcasm. This is taken care of by using a separate pipeline which will include models for sentimental analysis.

05 Conclusion


Customer engagement via quick automated solutions is one of the major factors enhancing the value of any organizational intellectual property. With virtual assistants being the focal point of many such automations, there is an obvious need to provide something close to best practice in the market. With NLP usability, it has now become possible to produce an organic conversation just like having two humans talking to each other. In this case, the assistant can immediately deliver solutions for the concerned user query/request. Thus, NLP used in Conversational artificial intelligence (AI) is a huge boost to the product performance metric, improving the overall end-user experience.



08 References



Mondal, Arnab (2019). Complete Guide to build your AI Chatbot with NLP in Python. Analytics Vidhya. Retrieved from

 <https://www.analyticsvidhya.com/blog/2021/10/complete-guide-to-build-your-ai-chatbot-with-nlp-in-python>

Lawton, George (2021). 5 reasons NLP for chatbots improves performance. TechTarget. Retrieved from

 <https://www.techtarget.com/searchenterpriseai/feature/5-reasons-NLP-for-chatbots-improves-performance>

Krayewski, Kaila (2022). How NLP Chatbots work. Ultimate. Retrieved from

 <https://www.ultimate.ai/blog/ai-automation/how-nlp-text-based-chatbots-work>

Jassova, Barbora (2021). Share of customers who have used to engage with companies in the United States as of 2019, by Industry [image]. Landbot. Retrieved from

 <https://landbot.io/blog/chatbot-statistics-compilation>

Author



Rohit Vishwakarma

Data Engineer, NAUT,
LTIMindtree

Rohit Vishwakarma is an AI enthusiast with knowledge of Python and Natural Language Processing in general. With a current experience of 2 years in IT, he is passionate about chatbots and has been developing complex virtual assistants since his graduation. He is currently a part of the NAUT Solution Design Group, where he handles the virtual assistant component deliverables alongside being responsible for DevOps strategies in the team.

LTIMindtree is a global technology consulting and digital solutions company that enables enterprises across industries to reimagine business models, accelerate innovation, and maximize growth by harnessing digital technologies. As a digital transformation partner to more than 750 clients, LTIMindtree brings extensive domain and technology expertise to help drive superior competitive differentiation, customer experiences, and business outcomes in a converging world. Powered by nearly 90,000 talented and entrepreneurial professionals across more than 30 countries, LTIMindtree — a Larsen & Toubro Group company — combines the industry-acclaimed strengths of erstwhile Larsen and Toubro Infotech and Mindtree in solving the most complex business challenges and delivering transformation at scale. For more information, please visit www.ltimindtree.com.