#Research

Integrate response retrieval models in assistants built with Rasa

Daksh Varshneya | Sep 12, 2019 | 6 min read

Here are some tutorials that can help you get started with Rasa. ✗

You can use Rasa NLU to build assistants in any language you want! ✗

The supported python versions are 3.5, 3.6 and 3.7! We recommend using 3.6!

Wait, I have a question. What python version is needed to install Rasa?

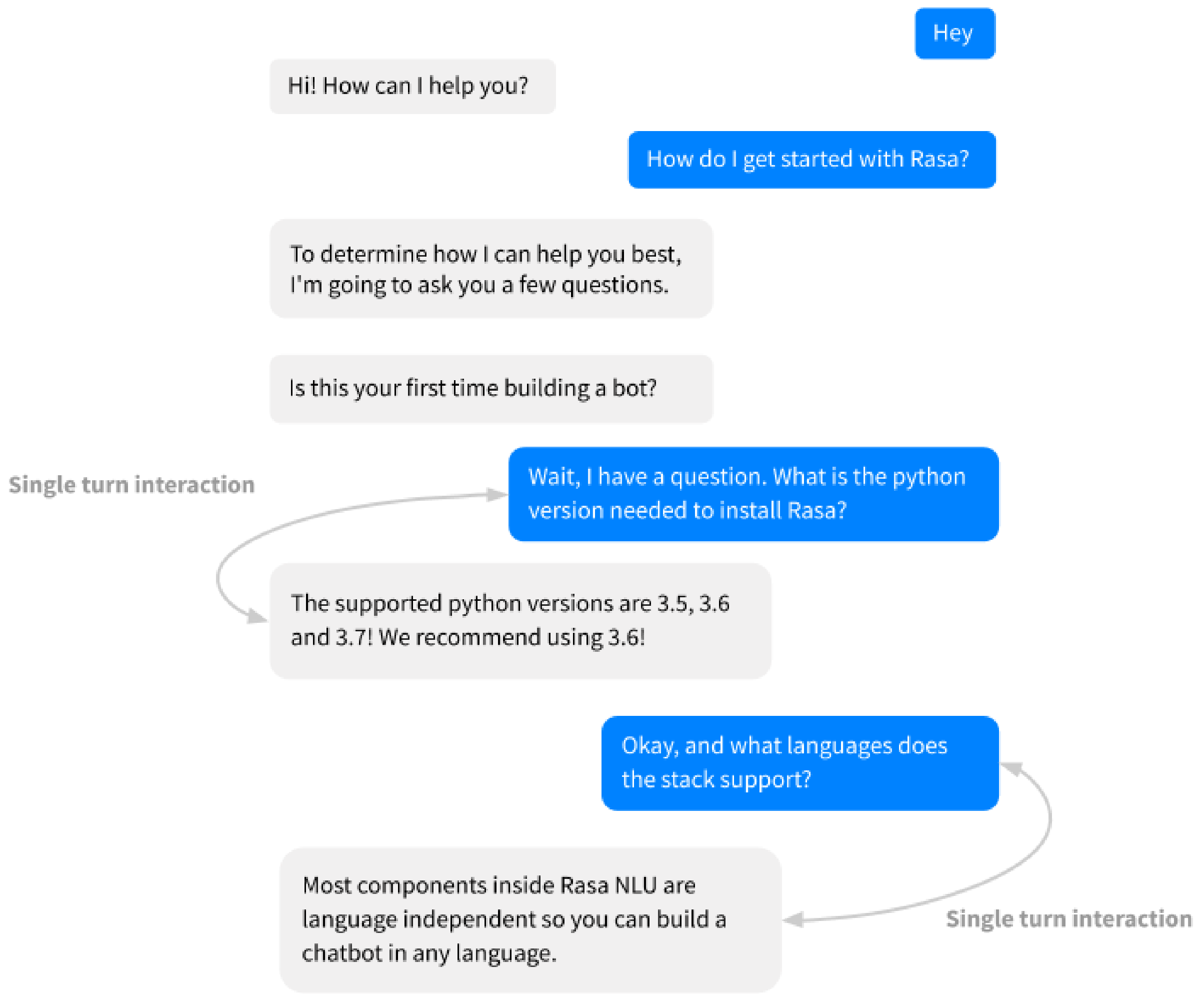
The supported python versions are 3.5, 3.6 and 3.7! We recommend using 3.6!

In this blogpost, we introduce a new experimental feature that adds retrieval-based response selection to Rasa. Rasa 1.3.0 introduces the Retrieval Action and the ResponseSelector NLU component. Both these components coupled together make it easier to handle specific dialogue elements like small talk**,** chitchat, FAQ messages, and other single-turn interactions in a simple manner. In short, this blogpost delivers three main ideas:

3. Building a model to select an appropriate assistant response in single-turn interactions.

Retrieval Actions

Retrieval actions are designed to make it simpler to work with FAQs, chitchat, and other single-turn interactions. By single-turn, we mean that your assistant should always respond the same way, regardless of what happened in previous interactions. Let's take a look at an example conversation:



Hey

Hi! How can I help you?

How do I get started with Rasa?

To determine how I can help you best, I'm going to ask you a few questions.

Is this your first time building a bot?

Wait, I have a question. What is the python version needed to install Rasa?

The supported python versions are 3.5, 3.6 and 3.7! We recommend using 3.6!

Okay, and what languages does the stack support?

Most components inside Rasa NLU are language independent so you can build a chatbot in any language.

When a user asks Sara (our demo bot) for the recommended python version. Sara

```
## Ask Python version
* ask_faq_python_version
    - utter_ask_python_version

## Ask languages supported in Rasa
* ask_faq_languages
    - utter_ask_languages

...
```

If you use a retrieval action, you just need one story!

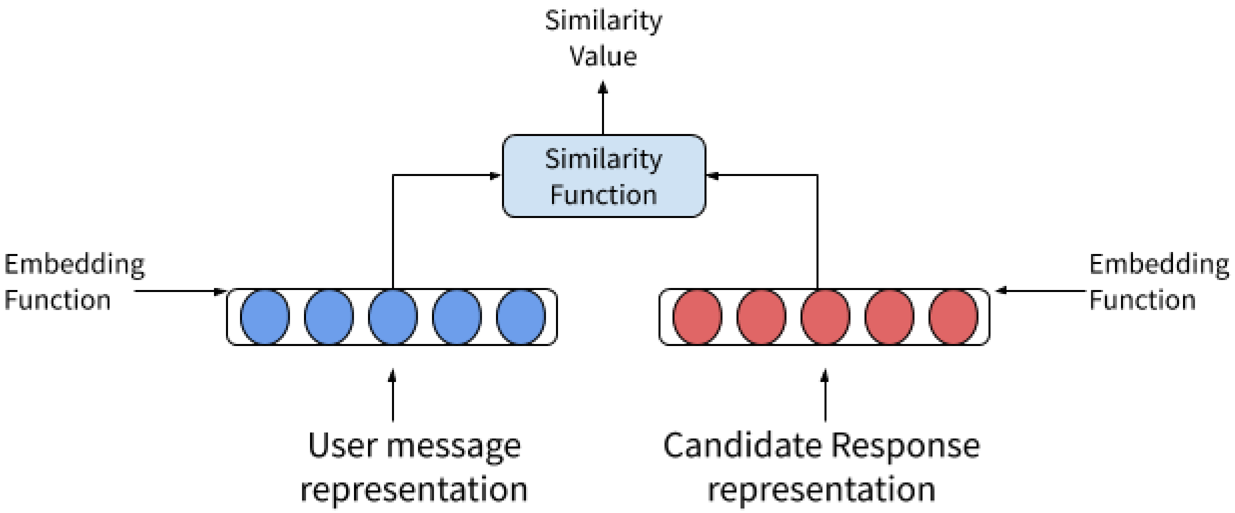
```
## Some question from FAQ
* ask_faq
    - respond_ask_faq
```

So what’s changed? All FAQ-related intents are grouped into one **retrieval intent** and are *responded* to by a single `respond_ask_faq` action. This makes it easier to treat all FAQ messages in the same manner, with a single retrieval action irrespective of their specific intent.

Since responses to such intents do not depend on previous messages, we do not need a sophisticated core policy to predict the corresponding retrieval action. But since there is a single retrieval action, we need to build a machine learning model to select the most appropriate response from all candidate responses for that action. How do we do that?

Supervised Response Selector

- Computing a learnt embedded representation separately for each, by passing the bag of word representations through densely connected layers.
- Applying a similarity function to compute the similarity between user message embedding and candidate response embedding.
- Maximizing similarity between correct user message and response pairs and minimizing the similarity for wrong pairs. This acts as the optimization function for the ML model during training.
- Matching the user message for similarity against all candidate responses at inference time and selecting the response with the highest similarity as the assistant's response to the user message.



This is very similar to how the EmbeddingIntentClassifier works, with the main difference being that the intent is replaced with the actual text of the response.

The component should be preceded by a tokenizer, featurizer and an intent classifier for it to process the incoming message. You can include the component in your NLU pipeline configuration as:

```
  - name: "EmbeddingIntentClassifier"
  - name: "ResponseSelector"
```

Training Data

Let’s take a look at the training data.

Retrieval Intents

Our retrieval intent `ask_faq` can have the following NLU examples -

```
 ## intent: ask_faq/python_version
- What version of python is supported?
- What version of python should I have to install Rasa
package?

## intent: ask_faq/languages
- Does Rasa support Chinese?
- What languages does Rasa stack support?
```

There are two related intents: `ask_faq/python_version` and `ask_faq/languages`. The intent classifier will group these intents into a single retrieval intent, `ask_faq`. Only the ResponseSelector cares about the difference between `ask_faq/languages` and `ask_faq/python_version`.

Response Phrases

The actual response texts are now part of the training data, so these will not be in your domain file. This is an important difference between responses for retrieval actions and

```
 ## FAQ python version <!--name of story-->
* ask_faq/python_version
    - Rasa currently supports python 3.5, 3.6 and 3.7! We
recommend using python 3.6.

## FAQ supported languages <!--name of story-->
* ask_faq/languages
    - Most components inside Rasa NLU are language
independent so you can build a chatbot in any language.
```

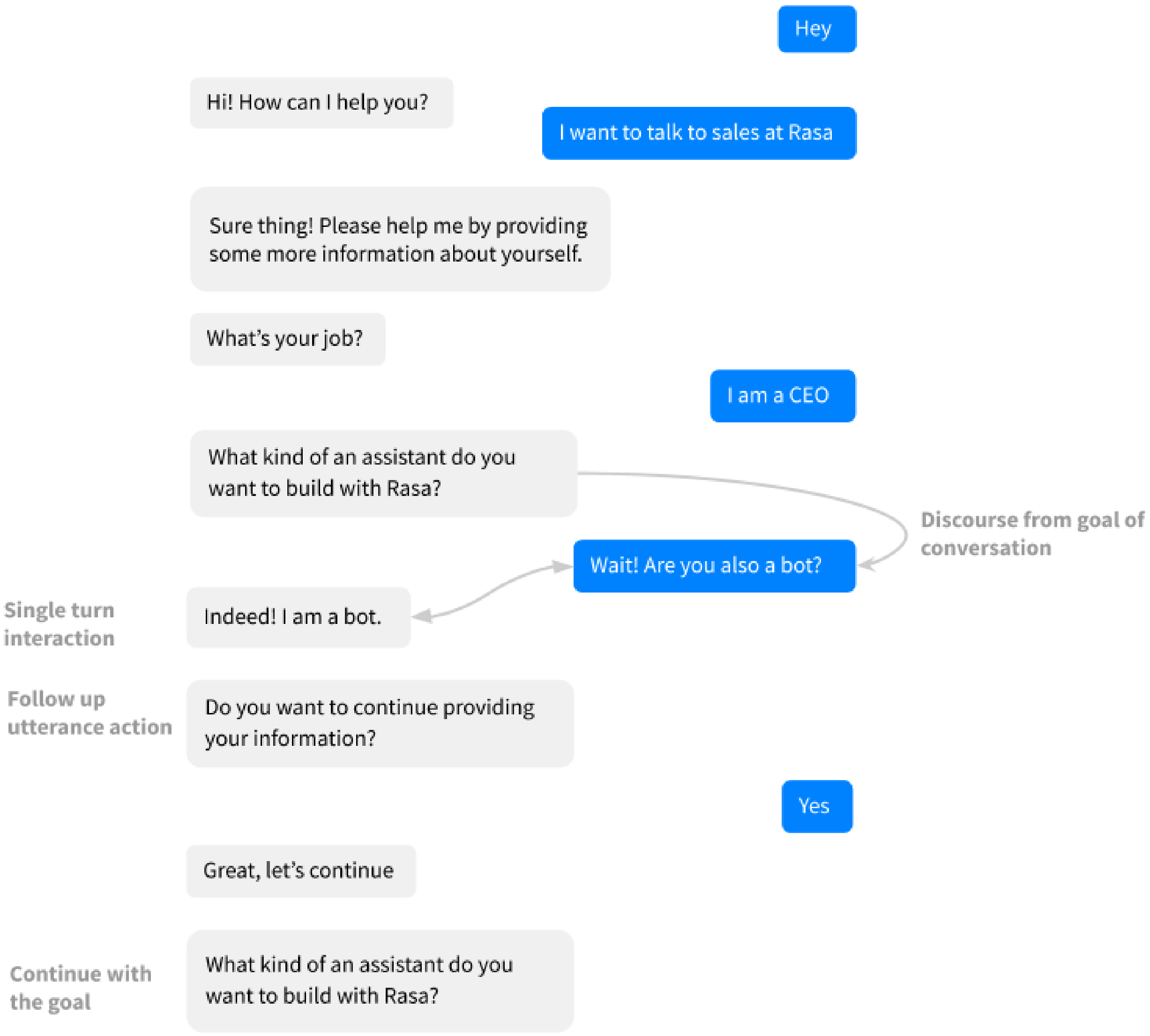
It is **mandatory** to have these response phrases in a separate file (you can call it responses.md, for example) and not in the same file as your NLU training data. The separate file can still be in the same folder containing other data files for NLU training.

Retrieval actions

Rasa uses a naming convention to match the names of retrieval intents like `ask_faq` to the correct retrieval action. The correct action name in this case is `respond_ask_faq`. The prefix `respond_` is required to identify it as a retrieval action. You should add retrieval actions to your domain file just like any other action. There are two ways to trigger these actions:

1. If you always want to just respond and then listen for the next user utterance, use the MappingPolicy to map retrieval intents to corresponding actions in the domain file:

```
intents:
    - ask_faq: {triggers: respond_ask_faq}
```



Hey

Hi! How can I help you?

I want to talk to sales at Rasa

Sure thing! Please help me by providing some more information about yourself.

What's your job?

I am a CEO

What kind of an assistant do you want to build with Rasa?

Wait! Are you also a bot?

Indeed! I am a bot.

Do you want to continue providing your information?

Yes

Great, let's continue

What kind of an assistant do you want to build with Rasa?

Discourse from goal of conversation
Single turn interaction
Follow up utterance action
Continue with the goal

The story for learning the above conversation would look like:

```
 * greet
     - utter_greet_user
* contact_sales
     - utter_moreinformation
     - sales_form
     - form{"name": "sales_form"}
```

```
    - sales_form
    - form{"name": null}
```

Overall, we tried simplifying the training data format on two primary factors:

1. Developer Experience: The training data format should be as intuitive as possible for developers and not extremely different from the existing training data format.
2. Logical Sense: All elements of training data should conform to the existing logical constructs of elements inside Rasa -- for example, training data should not be inside the domain file.

Playing a bit more with Response Selector

Rasa allows you to have multiple retrieval intents and correspondingly multiple retrieval actions as part of your assistant. In this case you have two options:

1. You can build one shared response selector model which would be trained on user utterance and response utterance pairs across all retrieval intents of your assistant. In this case, you do not need to define the `retrieval_intent` parameter in the configuration of the Response Selector:
2. You can build a specific response selector model for each retrieval intent. Each model would be trained on user utterance and response utterance pairs grouped under that retrieval intent. Hence, the number of response selector components in the NLU configuration should be the same as the number of retrieval intents in the training data. To do this, use the `retrieval_intent` parameter in the configuration of each Response Selector component to define the corresponding retrieval intent:

```
pipeline:
    retrieval_intent: chitchat
    …… # other architectural parameters
```

The choice between building a specific or a shared response selector model is use-case driven. If utterances in a particular retrieval intent are very domain specific, for example FAQ-related questions, it might not make much sense to learn a shared embedded representation with generic words coming from intents like `chitchat` and `greeting`. Your specific model may even perform better with different set of parameters than the ones used for the shared model. For example, the training may improve when using balanced batching as the batching strategy if you have much more data for some FAQs than for others.

We wanted developers to have this flexibility to try and test what works best for them. In our experiments, building a shared response selector model for all retrieval intents yielded similar results when compared to having individual models for each of them. If you observe different results please share them with us on the forum.

Why is the feature experimental?

With the introduction of response selector and retrieval actions, we came up with a new approach to tackle single-turn interactions. The training data format doesn't completely support end-to-end training but is still a step in that direction. Also, the response selector component lies at the intersection of Rasa NLU and Core. Although we believe end-to-end training is an exciting area to progress in, we want to receive enough feedback from the community on the overall developer experience, performance of the model, and the functionality itself before we take it further. Hence, we plan to keep the feature experimental for now. This means that the functionality may be changed or removed based on the feedback we receive.

Conclusion

keep us posted on how it goes on the forum. We are excited to see what new interesting use cases you build with this component!

Author

Daksh Varshneya

Machine Learning Researcher at Rasa

Join our Newsletter

Stay up to date with the latest news from the Rasa community

Email Address

Email

Follow us

Featured posts

1 **Introducing DIET: state-of-the-art architecture that outperforms fine-tuning BERT and is 6X faster to train**

2 **Rasa NLU in Depth: Part 1 – Intent Classification**

Tag cloud

#chatbots #community updates #customer stories #entities

#events #healthcare #hyperparameter #industry #intents

#machine learning #masterclass #nlp #open source

#product update #rasa core #rasa nlu #rasa x #research

#response selection #tutorials

1 Superhero Spotlight: Xiaoquan Kong

2 Connect in Fewer Steps: What’s New with Integrated Version Control

3 The Rasa Masterclass Handbook: Episode 12

4 Connect Your Rasa AI Assistant to Amazon Alexa

5 Using Conversation Tags to Measure Carbon bot’s Success Rate

Unpacking the TED Policy in Rasa Open Source

Train on Larger Datasets Using Less Memory with Sparse Features

Tanja Bunk Jan 28, 2020

Product

Why Rasa

Features

Support

Docs

Plans and Pricing

Compare Plans

Enterprise

Solutions

Enterprise Operations

Customer Service

Lead Generation & Sales

Financial Services

Healthcare

Voice

Case Studies

Industries

Healthcare

Insurance

Banking

Telecom

Travel & Transport

Community

Join the Community

How to Contribute

Community Showcase

Forum

Company

About Us

Careers

Our Mission

How we make money

Research

Blog

Contact Us