QuickHelp's Knowledge Script (QKnows 0.2, BETA)

Introduction

The QuickHelp's Knowledge Script (QKnows) is a Natural Language Processing system created with simplicity, transparency and ease of adoption in mind. The QKnows is a script written in Knowledge Entities (Knowets). Knowets are scripted data sets engaged to 'train' the chatbot created on the QuickHelps Talking Website Dashboard (Please learn more about Talking Website here: https://github.com/SYLMultimedia/quickhelp_TW).

Philosophy

In view of highly developed NLP systems that are already in use, QKnows is a bold attempt to create a yet 'simpler' alternative for 'non-technical developers'. The end point is to eventually achieving an application-specific language that enables almost anyone to train a chatbot on the QuickHelp System. In our thought is also a possibility of departure from a black box approach presented by more advanced NLP backend.

Knowets

Knowets are made up of Requests, Responses, Flows, Buttons, Links, Gestures, Images, Moods and Actions.

The QKnows algorithm allows users' requests to be understood by matching them with the most suitable Knowets. The matching does not need to be verbatim or follow any particular order.

When the Knowet does not exist or the pre-configured sensitivity is too low, the chatbot returns a negative narrative like "Sorry, I do not understand you".

The Chatbot's sensitivity feature may be configured on the dashboard for premium users.

Structure/Syntax of Qknows

The QuickHelp Knowledge Script (Qknows) is written with Knowledge Entities (Knowets). The script can be written manually, uploaded or simply generated with the dashboard. Also, the QuickHelp's Website Reader (QWR) reads to Qknows files which may easily be transferred and reused.

1. How to Write Knowets

Every knowledge entity (knowet) ends with an '%'.

Knowets are not case sensitive.

Every Knowet is identified and named by its Request. E.g How are you@Great!% is simply named 'how are you' knowet.

Every knowet is started an intent or Request (RQ) and followed by a Response (RS), separated by an '@' e.g.

Knowet: How are you@Fine!% .

The Request is “How are you “and the Response is “Fine!”

2. Flow

A flow is a way of linking one knowet to another to achieve a smoother and seamless conversation.

After the response (RS), introducing ANOTHER single '@' generates what is called a ‘flow’.

e.g.

Knowet: Help me@Why?@Thanks%.

The Flow in this example is ‘Thanks’.

When the flow is set, the Qknows automatically fires another request. In the example above, the bot responds with a 'why' and flows into another Knowledge entity (knowet), seeking the appropriate response for the word "Thanks", if the knowet ‘thanks’ exists.

Example

Knowet 1: Help me@You need help?@I need help%
Knowet2: I need help@What kind of help do you need?%

3. Story

A Qknows Story is a collection of Knowets linked together by flows. Stories are great for sustained or continuous responses to requests. For example, you can use a story to explain a concept using multiple Knowets.

Example

Knowet 1: Help me@You need help?@I need help%
Knowet2: created for that purpose@I was created exactly for that purpose.%
Knowet3: I need help@What kind of help do you need?@created for that purpose%

RQ: Help me
RS: You need help?
RS: What kind of help do you need?
RS: I was created exactly for that purpose.

Explanations: Note that the chatbot has a flow from one knowet to another by throwing the next flow and finding the next corresponding knowet. Note also that the knowets do not have to follow any particular order.

4. Multiple responses

Knowets may also be written to give multiple successive responses. Multiple responses are handled by using the '#' in between responses (RS) e.g How are you @Fine#Thank you!%

If you desire, you may also introduce a flow after multiple responses. e.g How are you@Fine#Thank you@your age%. Note that the 'your age' knowet must have been created for the flow to work.

5. Varying responses

Varying responses make your Chatbots less stupid .ie. smarter. When asked the same question repeatedly, the chatbot simply offer a different answer based on the intent or request.

NB: By default, the chatbot detects repetition and will not respond. This can be disabled on the dashboard is not desired if you are on Premium plan.
The Qknow script allows you to have varying responses for a simple request. This is achieved by introducing the '|' between the knowledge element.

Example

Where are you?@I am in Lagos | I am not around. %

This example suggests 2 variations of responses i.e 'I am in Lagos' or 'I am not around', randomly.
NB: This feature is only available to premium users.

6. Sentiments Detection

The QKnows is enabled to identify the sentiment of requests. When it detects a high or low sentiment, it looks for a matching Knowet, makes the appropriate response but also throws a high or low sentiment response, via a flow. Therefore the high and low sentiment knowets must also be pre-defined.

Example

Knowet1:What an idiot you are@Wao#Idiot is a strong word!%
Knowet2:low_knows@now you are making me very sad.% (predefined on dashboard)
RQ: What an idiot you are!
RS: Wao, Idiot is a strong word!
RS: Now, you are making me very sad.

Explanation: Qknows picks the sentiment of the request as low, and adds a ‘low knows’ flow, automatically. This also happens when it detects a happy or high spirited request like “You are smart!”.

7. Moods

Moods are ways of suggesting how your chatbot should respond per knowet. By default, the QKnows is made up of 3 mood templates: Low, Normal and High.

Premium users will be able to configure more variants.

When the mood is not specified, the chatbot defaults to Normal. The High mood is reflected with a highlighted text response. For now, the low mood is reflected by a coloured text response, like red, orange or brown.

Example

How are you@Fine#Thank you@@@mood:high %

This knowet example will expect a ‘How are you’ request to return ‘Fine’, ’Thank you’ in successive responses, with Highlighted texts, suggesting a great mood.

Images, links, gestures are also handled similarly (Please see the Knowet reference for more examples)

8. Complex requests

Complex requests are requests that are made up complex sentences i.e compound sentences that have two or more concepts, but joined together by punctuations. e.g ‘I am home. Have you cooked?’

Qknows automatically detects complex requests and breaks them down into simple individual requests, treating each component as an individual knowet. For now, Qknows does not relate each of the independent responses.

NB: This feature is only functional for premium users.

9. Contexts

Although this is still anon going development and not yet perfected, Qknows handles context by identifying key entities within the requests. For example, a request like “Do you have some candy?” will automatically triggers the context ‘candy’. Further requests like “where did you buy it?” will be reconverted to “when did you buy candy” before firing the appropriate knowet.

Example 1, for the following given knowet:

Knowet : Help me bring my bag@Ok#Where is it?%
RQ: It is in the bathroom!

Explanation: The RQ will be reconverted to “bag is in the bathroom” and that fires the ‘bag is in the bathroom” Knowet if it exists.

Example 2, for the following given knowet:

Knowet : I Do you love dancing?@Yes I do%
RQ: What about Swimming?

Knowet : I Do you swim?@Yes I do%

Explanation:

QKnows will identify the entity ‘swimming’ and the Context trigger “what about” and the Request will be reconverted to “Do you love Swimming” and that fires the ‘Do you love swimming” Knowet if it exists.

Note that Knowet ‘Do you love swimming’ does not exist but the closest knowet fires ‘Do you swim?’.

10. Repeat requests (e.g Say that again)

The QKnows allows your chatbot users to make a polite request for repetitions of previous answers. E.g

Knowet: Are you smart @Yes, I am trying%

RQ: Are you smart?
RS: Yes, I am trying.
RQ: What did you say?
RS: I said Yes, I am trying.

Explanation: The QKnows detects the intent ‘what did you say ‘and recognizes it as a request for repetition.

11. Agreements, yes/no responses

The Qknows is also able to throw a pre-flow when it detects a yes or no response. It does this by prefixing a yes or no (or synonyms) to original request and throwing is as a flow.

Example

Knowet1: What is the meaning of KNOWET@KNOWET means knowledge entity@ just knowing this%
Knowet2: Just knowing this@ Are you just knowing this?!%
Knowet3: **Yes** Just knowing this @ Oh!, Ithought you were following me!%

RQ:What is the meaning of KNOWET?
RS: KNOWET means knowledge entity.
RS: Are you just knowing this?!
RQ:Yes
RS:Oh!, Ithought you were following me!

Explanations:

Note that Knowet 1 has a flow 'just knowing this'. This flow will search for knowet2 and fire the response 'Are you just knowing this?!'. If a 'yes' is returned, Knowet3 is triggered if it exists. If not, it will default back to Knowet 2.

Caution: This might create an unwanted loop so all Yes/No (polar) question knowets must always be followed by a corresponding yes or no knowets for the appropriate reponses.

12. Wikipedia

The QKnows sometimes defaults to Wikipedia for unknown requests thereby creating an undesirable response.

13. Knowets Reference

RQ@@@how are you?% Creates a request call to action button. e.g How are you@@@Find out here%.
RQ@Great!@@how are you?% Creates a request call to action button after a response. e.g How are you@@@Find out here%.
RQ@@@how are you?(link:'http://yourурl') % Creates a button. e.g How are you@@@Find out here(link:'http://yourурl')%.
RQ@Great!@@mood:high % Suggests a response 'Great' with a high mood.
RQ@Bad@@ mood:low % suggests a response 'Bad' with a low mood.
RQ@Get lost@@ gesture:dock% Suggests a response 'Get lost' with a gesture. In this case, the gesture is docking i.e the AI agent minimizes.
RQ@I am Typing!@@ action:type:Sugar% Suggests a response 'I am typing' with an action. In this case, the action is to type the word 'Sugar'.RQ@This is my face@@ image:'imageurl.jpg'% Suggests a response 'This is my face' with the image displayed referencing the url 'imageurl.jpg'.

Further Development

Even by the developers, the QKnows is a still a very naïve or over-simplified attempt at NLP. It is also still adjudged complicated compared to the original dream of having a true scripting language for non-technical users. Also, initial testing is proving that the Qknows is still far from being consistent with intended responses.

However, the QuickHelp development team has proposed Qknows as it is, while it works on the following:

- Dynamic Entities
- More Context Triggers
- Stemming and Lemmalization
- Parts of Speech Tagging
- Shallow Parsing
- Constituency and Dependency Parsing
- Emoticons and shorthands
- Summarization
- Multilanguage Support

Remember that to use Qknows, you needto sign up for QuickHelps TalkingWebsite here: https://widget.quickhelp.com.ng/dashboard/signup.php

For Suggestions, criticism, feedbacks or contributions, please send an email to Tosin at qknows@quickhelp.com.ng

Development Team

Oluwatosin Odubela
Lead Programmer
twitter: @tosinloluwa

Akin Akinseinde
Data Scientist

Olayinka Odubela
Linguistics

Qknows Copyright 2019. All rights reserved.