

QuickHelp's Knowledge Script (QKnows 0.4, BETA)



Introduction

Philosophy

What's new?

Prerequisite

Knowets

Structure/Syntax of Qknows

1. How to Write Knowets
2. Flows
3. Story
4. Multiple sequential responses
5. Varying responses
6. Synonyms/Varying Requests
7. Sentiments Detection
8. Moods
9. Complex requests
10. Contexts
11. Repeat Requests
12. Agreements (yes/no) requests
13. Wikipedia
14. Knowet Reference

Further Development

Development Team



Introduction

The QuickHelp's Knowledge Script (QKnows) is a Natural Language Processing (NLP) system created originally for use on the QuickHelp's [TalkingWebsite](#) dashboard. It was developed with simplicity, transparency and ease of adoption in mind. QKnows is written in **Knowledge Entities** (Knowets in short). Knowets are scripted data sets engaged to 'train' Chatbots 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 and advanced 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. The Qknows is being imagined also, as a possibility of departure from a black box approach presented by more advanced NLP backend.

What's new?

The fourth release of Qknows (0.4, beta) is a slight improvement on version 0.3. For instance, Qknows now allows for multiple varying requests per Knowet. In addition, the hyper link script has now been depreciated as hyperlinks are now clickable by default. Also, varying responses and requests are now available for non-premium users.

Prerequisite

Writing Qknows requires that an account is opened on QuickHep's TalkingWebsite. A new account may be opened with the following URL: <https://widget.quickhelp.com.ng/dashboard/signup.php>

After opening an account, the '**Chatbot Trainer**' link opens a text field where the Qknows is written, published and updated.

Knowets

Knowets **are single-lined data entities** usually composed of Requests, Responses, Flows, Buttons, Links, Gestures, Images, Moods and Actions. They form the building block for Qknows.



QKnows algorithm allows users' requests to be 'understood' by matching requests 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 Chabot'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

- a. Every knowledge entity (Knowet) ends with an '%'
- b. Knowets are not case sensitive.
- c. Every Knowet is identified and named by its Request. E.g How are you@Great!% is simply named 'how are you' Knowet.
- d. Every Knowet is started an Intent or Request (**RQ**) and followed by a Response (**RS**), separated by an '@' sign 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 (triggers) another request from within the script. 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%*

Results

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 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 exist for the flow to work.

5. Varying Responses

The QKnows script allows you to have varying responses for a simple request. This is achieved by introducing the '|' sign in between the knowledge element.

Varying responses make your Chatbots less repetitive .i.e. 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.



Example

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

This example suggests two variations of responses i.e 'I am in Lagos' or 'I am not around', which are randomly presented.

NB: This feature is now available for non-premium users.

6. Synonyms/ Varying Requests

Similarly, the Qknows have the capacity to enable multiple requests per Knowet, in form of synonyms, similar clauses or joint intents. This allows for setting varying requests that may trigger just a single response.

Example

Knowet: ***Who are you? | Identify yourself @I am QuickE#a Chatbot%***

Explanations: The requests 'Who are you' or 'Identify yourself' when triggered will give the same response 'I am QuickE, a Chatbot'.

7. 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 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)

Results

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!".



8. 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).

9. 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.

10. Contexts

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. Please note that this is still an ongoing development and not yet perfected.

Example 1

Given the following knowet and requests respectively:

Knowet : ***Help me bring my bag@Ok#Where is it?%***

Knowet : ***Bag in the bathroom@Alright then#I am going for it right away!%***



RQ: Help me bring my bag!

RS: Ok, Where is it?

RQ: It is in the bathroom! (Converted to "Bag in the bathroom")

RS: Alright then, I am going for it right away!

Explanation: The second RQ will be reconverted to "bag is in the bathroom" and that fires the 'bag is in the bathroom' Knowet.

Example 2

Giving the following knowets and requests:

Knowet : Do you love dancing?@Yes I do%

Knowet : Do you swim?@Yes I do%

RQ: Do you love dancing?

RS: Yes I do

RQ: What about Swimming?

RS: Yes I do

Explanation: QKnows will identify the entity 'swimming' and the Context trigger "what about". Therefore, 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?'.

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

The QKnows allows your Chatbot users 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.

12. 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 it 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! I thought you were following me!%*

Results

RQ: What is the meaning of KNOWET?

RS: KNOWET means knowledge entity

RS: Are you just knowing this?

RQ: Yes

RS: Oh! I thought 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 Responses.

13. Agreements, yes/no responses

The Qknows is developed to sometimes default to Wikipedia for unknown requests. This serves to enrich unknown conversations especially when making requests about the definition or description of common or popular entities.

14. Knowet 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@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

The developers know that QKnows is still a very naïve or over-simplified attempt at NLP. From early feedbacks, it is also still adjudged generally 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
- Tag enabled context Triggers
- Stemming and Lemmalization
- Parts of Speech Tagging
- Shallow Parsing
- Constituency and Dependency Parsing
- Emoticons and shorthand
- Summarization
- Multilanguage Support

Some immediate pursuits include:

1. Development of a Restful API
2. Interfacing with Tensorflow

Remember that to use Qknows, you need to sign up for QuickHelps TalkingWebsite here:

<https://widget.quickhelp.com.ng/dashboard/signup.php>

For Suggestions, criticism, feedbacks, donations 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.

