QuickHelp Knowledge Script

Qknows (0.6) BETA

User Guide | 25092024 | QuickHelp Nigeria
https://quickhelp.com.ng/qknows/

Table of Content

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).

Current Features

Reusable Simple Sentences

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

Key features

Aside from simple intent triggers (or prompts), the Qknows framework supports features such as call-to-action buttons responses, audio and video responses, speech recognition, **sentiment analysis and mood detection**, named entity templates, automatic text reader as well as text to speech capabilities.

Access and Integration

QKnows is by default, integrated on the QuickHelp's ***Talking Website*** Dashboard. However, in addition to its simple Script integration option, the framework can connect with Google's DialogFlow, OpenAI's GPT and Hugging Face . Qknows also has a plugin created for Wordpress.

Custom Templates as Responses

QKnows also supports call-to-action responses (CTA), Image responses, Video responses and YouTube Videos as responses.

Built in Named Entity Recognition

QuickHelp adopts a custom-built algorithm to detect and recognize entities within a prompt, which may be harvested for interaction with external APIs for more intuitive and engaging integrations.

Philosophy

In view of highly developed and advanced NLP systems, QKnows is a bold attempt to create a yet simpler alternative especially for non-technical developers. The end point is to eventually achieve an application-specific language that enables almost anyone to train a Chatbot on the QuickHelp ecosystem. The Qknows is being imagined also as a possibility of departure from a black box approach presented by more advanced NLP backend. Qknows also portends opportunities for explainable AI by adopting these principles:

- **Human-in-the-Loop**: Building controllability into AI powered conversation agents i.e. the ability for users to adjust and influence the behavior or outputs of the AI system in real time, ensuring that decisions align with human intent and desired outcomes. This principle emphasizes empowering users to make adjustments based on their understanding of the system's explanations.
- **Transparency**: Making AI powered conversation agents and their decision-making processes clear.
- **Interpretability**: Allowing humans to understand why the AI made a specific decision.
- **Trustworthiness**: Building confidence in the system by providing understandable insights.

What's new?

Custom Templates as Responses

QKnows now supports call-to-action responses (CTA), Image responses, Video responses and YouTube Videos as responses.

Built in Named Entity Recognition

QuickHelp adopts a custom-built algorithm to detect and recognize entities within a prompt, which may be harvested for interaction with external APIs for more intuitive and engaging integrations.

Prerequisite

Writing Qknows requires that an account on QuickHep's TalkingWebsite. An account may be created with the following URL: https://widget.quickhelp.com.ng/dashboard/signup.php. On opening of the account, the 'Chatbot Trainer' tab opens a workspace where the Qknows is written, published and updated. The Qknows may also be **written externally** and uploaded with a simple text file. The account creation requires that a **pre-prompt** is set for the Chatbot, narrated as "What does your chatbot do?".

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 the Qknows.

The 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 (tune) feature may be configured on the dashboard** .

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 Qknows Website Reader (QWR) reads to Qknows files which may easily be transferred and reused.

1. How to Write simple 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

   Example

   Knowet: How are you@Fine!%
   Whenever the request (RQ) is “How are we “and the response (RS) is “Fine!”

2. Multiple Responses

Knowets may also be written to give multiple successive responses. Multiple responses are handled by using the '#' in between responses (RS).

Example

How are we @Fine#Thank you!%

RQ: How are you?
RS(1): Fine
RS(2): Thank you!

3. Varying Responses

The QKnows script allows we to have varying responses for a simple request. This is achieved by introducing the '|' sign in between the knowledge element. Varying responses make our 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 if not desired**

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.

4. Flows

A flow is a way of linking one Knowet to another to achieve a smoother and seamless conversation. For instance, it allows for for uninterrupted and continuous generation of responses by an AI agent, **until interrupted**.

Example

how are you@I am fine#flow::I am fine::%
I am fine@What about you?#flow::what about you::%
what about you@Hope you are doing fine too?%

The Flow in this example is 'I am fine'.

When the flow is set, the Qknows automatically fires (triggers) another request from within the script. In the example above, to the request "how are you" the bot responds with a 'I am fine' and flows into another Knowledge entity (Knowet), seeking the appropriate response for the word "I am fine", if the Knowet 'thanks' exists. In this case the follow up response is "Hope you are you are doing fine too? "

Result

RQ: how are you?
RS: I am fine
RS: What about you?
RS: Hope you are doing fine too?

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, we can use a story to explain a concept using multiple Knowets.

Example

Knowet 1: Help me@We need help?#Flow::I need help::%

Knowet2: created for that purpose@I was created exactly for that purpose.%

Knowet3: I need help@What kind of help do we need?Flow::created for that purpose::%

Results

RQ: Help me

RS: We need help?

RS: What kind of help do we 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.

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 predefined.

Example

Knowet1: What an idiot we are@Wao#Idiot is a strong word!%

Knowet2: low_knows@now you are making me very sad.% (triggered automatically, predefined on dashboard)

Results

RQ: What an idiot we are!

RS: Wao, Idiot is a strong word!

RS: Now, we 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 "We are smart!".

8. Moods

Moods are ways of suggesting how our Chatbot should respond per Knowet. By default, then QKnows is made up of 3 mood templates: Low, Normal and High. Users can 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 we 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.

10. Contexts

Qknows handles context by identifying key entities within the requests. For example, a request like “Do we have some candy?” will automatically triggers the context ‘candy’. Further requests like “where did we buy it?” will be reconverted to “when did we buy candy” before firing the appropriate Knowet.

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

Given the following knowets and requests:

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

Knowet : Do we swim?@Yes I do%

RQ: Do we 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 we love Swimming” and that fires the ‘Do you love swimming” Knowet if it exists.

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

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

The QKnows allows our Chatbot users make a polite request for repetitions of previous answers.

Example

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

RQ: Are we smart?

RS: Yes, I am trying.

RQ: What did we say?

RS: I said Yes, I am trying.

Explanation: The QKnows detects the intent 'what did we 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 is as a flow.

Example

Knowet1: yes What is the meaning of KNOWET?@KNOWET means knowledge entity# Are you just knowing this?!%
Knowet2: yes What is the meaning of KNOWET@ Oh! I thought we were following me!%

Results

RQ: What is the meaning of KNOWET?

RS: KNOWET means knowledge entity

RS: Are we just knowing this?

RQ: Yes

RS: Oh! I thought we were following me!

13. Default Responses

The Qknows is developed to sometimes default to Wikipedia or Open AI for unknown requests, **depending on the dashboard settings**. This serves to enrich unknown conversations especially when making requests about the definition or description of common or popular entities.

14. **Named Entity Templates**

Qknows allows entity templates that work with an external text file that have the following json format:

[{"entity":"occupations","categories":["shoes"]},{"entity":"locations","categories":["ikeja"]}]

E.g

{occupations at locations}@beeni%

The above Knowet gives the response:

RQ: shoes at ikeja
RS: beeni

Explanations: the templates are replaced by the data in the categories

The actual request becomes

RQ: $occupations at $locations

The response may be verbatim or via an API to creatively generate responses based on the produced variables. In this case, have:

$locations and $occupations

15. Button Templates

Qknows now allows button template to generate CTA (click to Action) responses. The buttons, on click, makes request based on their labels.

Example

Where is the button@Ok#button::click me::%

Explanations

RQ: Where is the button
RS: (a button with the label click me)

16**. Image Template**: Qknows now allows image template to provide image responses. NB: The image must be in PNG format, and the extension should not be specified.

Example

show face@I love my job!#image::quicke::#I am very happy to help%

RQ: show face
RS: (the image quicke.png is loaded)

17. **Video Templates**: have created a video template to generate video responses. NB: The image must be in mp4 format, and the extension should not be specified.

Example

show videe video@Ok#video::videe::%

RQ: show videe video
RS: (the video videe is loaded)

18. **YouTube Templates**: have created a YouTube template to play YouTube videos as responses. NB: The YouTube video's URL must be specified

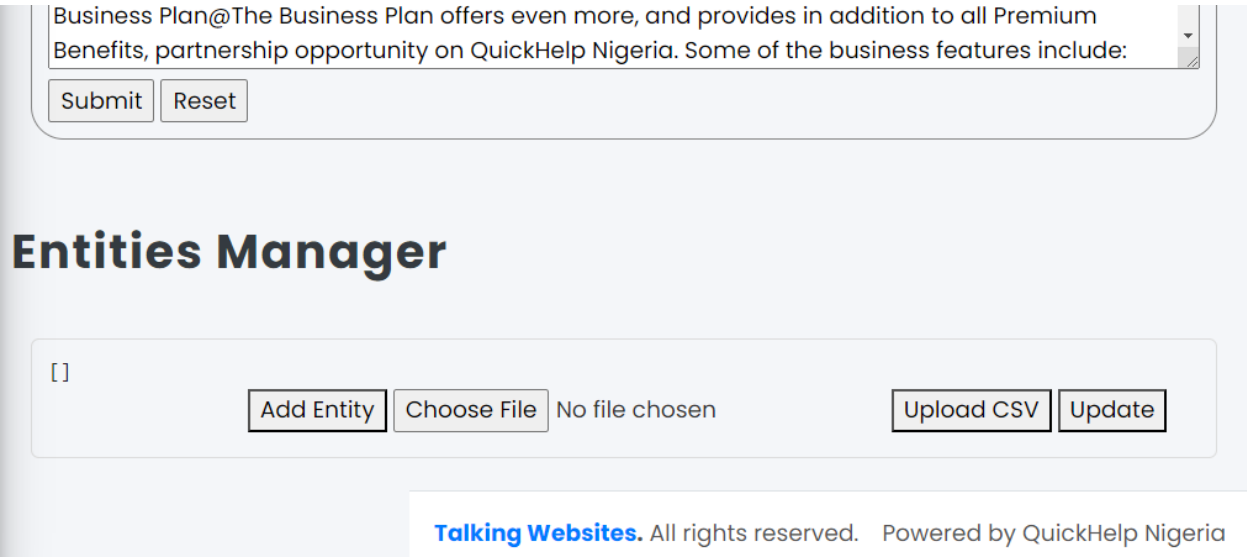
Example

Play video@
youtube::https://www.youtube.com/embed/3K0N871RjaY?si=L_cPkIjR9ycOxUmu::%

RQ: play video
RS: (the youtube video is loaded)

19. Data on Dashboard

The QuickHelp Dashboard may now allow the specification of Entities' templates by imputing or uploading data.



20. Switch

Qknows may now enable permissions to Switch from QuickHelp Native Agent to other chatbots created within the QuickHelp ecosystem. Qknows allows for a switch from one Chatbot agent's knowledge script to another.

Example

Chatbot 1 (Qknows1)

{Switcho to Calabar}@ %

RQ: Switch to Calabar
RS: You are attempting to switch over to Calabar now…

Chatbot 2 (Qknows 2)

Intro_name_request@You are welcome!%

RQ: intro_name_request (triggered by the switch function)
RS: You are welcome to Calabar Chatbot

Explanations

With the first Chabot (Qknows1), the Entity "switcho" have been defined to have words like: switch as part of its named entities so the request actually returns "switch to Calabar" which triggers the switch function. The chatbot switches to Calabar is there is any other chatbot on QuickHelp named Calabar.

The second Chatbot (Qknows2) has a predefined template called "Intro_name_request" which is usually adopted to welcome new users on the Qknows system and obtain basic information about the user like the name and email address.

//todo : combining Qknows

21. Knowet Reference Guide

1. Basic Knowet Structure

Format: `Request@Response%`

Example: `How are you@Fine!%`

2. Multiple Sequential Responses

Format: `Request@Response1#Response2%`

Example: `How are you@Fine#Thank you!%`

3. Varying Responses

Format: `Request@Response1 | Response2%`

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

4. Flows

Format: `Request@Response#flow::FlowTrigger::%`

Example: `how are you@I am fine#flow::I am fine::%`

5. Synonyms/Varying Requests

Format: `Request1 | Request2@Response%`

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

6. Sentiments Detection

Format: `Request@Response%` (with predefined sentiment Knowets)

Example:

```
What an idiot you are@Wow#Idiot is a strong word!%

low_knows@Now you are making me very sad.%
```


7. Moods

Format: `Request@Response#mood::MoodType::%`

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

8. Contexts

Format: Standard Knowet format (context is handled automatically)

Example:

```
Help me bring my bag@Ok#Where is it?%

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

9. Named Entity Templates

Format: `{EntityType1 at EntityType2}@Response%`

Example: `{occupations at locations}@beeni%`

10. Button Templates

Format: `Request@Response#button::ButtonLabel::%`

Example: `Where is the button@Ok#button::click me::%`

11. Image Template

Format: `Request@Response#image::ImageName::%`

Example: `show face@I love my job!#image::quicke::#I am very happy to help%`

12. Video Template

Format: `Request@Response#video::VideoName::%`

Example: `show videe video@Ok#video::videe::%`

13. YouTube Template

Format: `Request@youtube::YouTubeURL::%`

Example: `Play video@youtube::https://www.youtube.com/embed/3K0N871RjaY?si=L_cPkIjR9ycOxUmu::%`

14. Switch

Format: `{SwitchTrigger to ChatbotName}@%`

Example: `{Switcho to Calabar}@%`

Note: Some features like complex requests, repeat requests, and agreements (yes/no responses) are handled automatically by the Qknows system and don't require specific Knowet formats.

Further Development

The developers know that QKnows is a 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 and HuggingFace APIs

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

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

Strengths and Limitations of Qknows NLP Approach

The Qknows approach attempts to strike a balance between simplicity and functionality, making it accessible for many use cases. However, for more complex or large-scale applications, more advanced NLP systems might be necessary. Here are some strengths and limitations of the Qknows system:

Strengths

1. **Simplicity and Transparency**:
    - Easy to understand and implement, even for non-technical users.
    - Transparent process, allowing users to see exactly how responses are generated.
2. **Flexibility**:
    - Supports various response types (text, buttons, images, videos, YouTube).
    - Allows for context-aware responses and sentiment detection.
3. **Customization**:
    - Easy to create domain-specific chatbots with custom knowledge.
    - Supports multiple response variations for natural-sounding conversations.
4. **Integration**:
    - Can be integrated with more advanced NLP systems like DialogFlow, GPT, and Hugging Face.
    - WordPress plugin available for easy website integration.
5. **Human-in-the-Loop**:
    - Allows for easy adjustments and improvements to the chatbot's responses.
    - Supports the principle of explainable AI.
6. **Lightweight**:
    - Simple text-based format that's easy to store, transfer, and modify.
7. **Multi-modal Responses**:
    - Supports text, buttons, images, videos, and YouTube links as responses.

Current Limitations

1. **Limited Natural Language Understanding**:
    - Relies heavily on pattern matching as against deep language understanding.
    - May struggle with complex or ambiguous queries.
2. **Limited Context Handling**:
    - While it has some context awareness, it may not handle complex, multi-turn conversations as well as more advanced NLP systems.
3. **Limited Language Processing**:
    - May struggle with spelling errors, synonyms, or paraphrases not explicitly defined.
4. **Lack of Deep Understanding**:
    - Cannot truly understand or reason about the content; it's essentially a sophisticated pattern-matching system.
5. **Limited Ability to Generate Novel Responses**:
    - Responses are pre-written, limiting the system's ability to generate truly novel or creative answers to unforeseen questions.

Qknows Development Team

Oluwatosin Bukun-Joseph
Founder, Team Lead, CEO SYL Multimedia & QuickHelp Nigeria

Olayinka Odubela
Lead Linguist and Chatbot Analyst
Lecturer, Department of English Language, Obafemi Awolowo University, Ile-Ife

Aramide Adebesin
Lead Data Scientist
Mathematics Department, Obafemi Awolowo University, Ile-Ife

Other Contributors

Adebiyi Adeyanju | Data Scientist

Dr. Amole Abraham Olatide | AI/ML Engineer

Cynthia Sitati | Data Scientist

Davies Obiekea | Data Scientist

Effiong Edet | Data Scientist

Ejike Emeribe | Data Scientist

Ifeanyi Okonkwo | Data Scientist

Maximillah Ahuta | Project Manager

Oni Akinwumi Peter | Software Engineer

Ogundeji Gabriel | Software Engineer

Raji Sherifdeen | Software Engineer

John Benjamin | Software Engineer

Victoria Adewuyi | Assistant General Manager, SYL

Akingbehin Darasimi | Software Developer

Omidire Emmanuella | Software Developer

Qknows Usage License

For more information, please contact:

QuickHelp Nigeria

Email: admin@quickhelp.com.ng

Phone: +23408038615445