Moni.ai speech platform api

The restful Moni api interface is available on

https://api.moni.ai

Currently the following commands are supported

Login/Signup

A user can be initially logged in by *userid* and *password*.

When successfully logged in an *authtoken* is returned.

In order to perform actions as the logged in user this authtoken needs to be passed with every api call either as GET or as POST variable.

Userid and *password* are only accepted as post variables.

In order to signup a new user in addition to *userid* and *password* the variable *realname* must be provided. In this case a new user is created and the *authtoken* is returned.

Querying content

Content can be queried without logging in. In this case only content is retrieved that is not marked as private. The query string is passed using the GET variable *squery*.

The GET variable *rw* defines how the database stores questions and manipulates content.

Valid values for *rw* are:

0/not defined: no content is manipulated, content is only fetched read only

1: online services (wikipedia, yahoo answers, reddit) are queried in order to retrieve content

2: in addition to rw=1 the new question is stored in the database and it is remembered that the question has already been asked by the user

3: in addition to rw=2 allow the system to ask open questions sporadically and store the answer in the database.

Querying own content

In order to retrieve content that has been entered by the user himself an api call can be made with a GET/POST variable *mode* set to 2. For this the user has to be logged in first.

Querying questions

Open questions – that are questions with only bad answers – can be queried with *mode* set to 6. Open questions can be queried anonymously. Two additional GET variables *limitcount* and *limitoffset* are available to limit the search result.

If not open questions shall be queried but questions according to a given query string the additional variable *squery* can be passed containing the query string.

Querying answers

In order to query the list of available answers to a specific question the variables *question* must be provided. *Question* contains the id of the question for which answers shall be returned.

Saving new question/answer pairs

A logged in user can save a new question/answer pair in the database or add an answer to an existing question.

The following are required parameters:

mode: must be set to 1

question: a pattern that is stored as question in the database

or

questionid: an existing questioned as alternative to a question.

answertext: a pattern that is stored as answer to the question in the database.

The following arguments are optional:

actionIsPrivate: defines whether the answer pattern is private and will only be given to the user

action: defines an actionid (see list of valid actions below)

actionval: defines the value corresponding to the action

opt1: defines an optional argument needed for the given action

voting question/answer pairs

In order to vote up/down a question or a question/answer pair the following post variables are needed:

Vote: must be 1 (upvote) or 2 (downvote)

Cqaid: id of the question/answer pair

OR

question: id of the question to downvote (login required)

Deleting answers

A logged in user can delete own answers. The following parameters are required:

mode: must be set to 3

Either *answer* or *cqaid* or both must be given:

answer: the id of the answer (e.g. retrieved from answerid when querying own content with mode=2)

cqaid: the id of the connection between question / answer to be deleted (e.g. retrieved from cqaid when querying own content with mode=2)

Valid actions

Valid actions are the following:

0: no action

1: open url (*actionval* must be given)

2: an ifttt.com action is triggered (*actionval* defines the ifttt tag, *opt1* defines the sender email on ifttt.com)