BachelorThesis

This Project is part of my Bachelor Thesis.
The aim is to create a DSL to configure Dialogflow agents via a small configuration file, that is easily human readable and source-control firendly.
Among benefits provided by this approach are:

- Quick creation of Dialogflow agents via script file with IDE support
- Small Diffs in source control

This is developed alongside Ask your Repository where we are planning to use this DSL for the configuration of our agent.

Documentation:

Filetype: This language is associated with the .dfc file ending.
It compiles to plain JSON using the compiler you can find here

Syntax:

The simplest Agent possible consists of an Agent object with a language tag and no intents or entities.

```
Agent SampleAgent
    language en
```

Settings

From here on more OPTIONAL tags can be added:

description: A string describing the agent.
Defaults to an empty string

version: The version number of this agent in major.minor.micro notation.
Defaults to 1.0.0

ml_classification_threshold: The minimum confidence at which ML-classification should be trusted.
Defaults to 0.4

enable_logs, log_to_google_cloud, private, hybrid_match_mode: These are all keywords that if entered into the agent config will set a corresponding value and will default to false.
enable_logs: Log interactions to Dialogflow.
log_to_google_cloud: Save logs to google cloud, requires enable_logs to work. private: true if the "Private" option is selected in the agent settings, false if the "Public" option is selected.
hybrid_match_mode: Use the Hybrid (Rule-based and ML) mode for agents with a small number of

examples/templates in intents, especially the ones using composite entities. Use ML only mode for agents with a large number of examples in intents, especially the ones using @sys.any (default).

In code configuring all of these, and setting all the settings to true would look as follows:

```
Agent SampleAgent
    language en
    description 'This is a sample agent to show how the .dfc language is
specified.'

    version 0.1.1
    ml_classification_threshold 0.7
    enable_logs
    log_to_google_cloud
    hybrid_match_mode
    private
```

WebHook

Another Optional Setting with more detailed configuration is the WebHook setting:
It allows to set the agent to send requests to a webserver of your choise via https. It is configured as follows:

```
Agent SampleAgent
    language en
    version 0.1.1
    description 'This is a sample agent to show how the .dfc language is
specified.'

    Webhook
        active
        url 'https://sample.url.com/webhook'
        headers
            'key1' : 'value1'
            'key2' : 'value2'
```

If the keyword active is not included, the webhook will still be configured but it will not be used.

Entities

All the default entity-types in Dialogflow can be used without configuration. Defining a custom entity-type works as follows:

```
Agent SampleAgent
    language en
    version 0.1.1
    description 'This is a sample agent to show how the .dfc language is
specified.'
```

```
    Type SampleEntityType
        values "Type1" ('Type1 Synonym1' 'Type1 Synonym2'), 'Type2', 'Type3'

    Type SampleEntityType2
        dynamic
```

The keyword dynamic tells the compiler, that you will set this entity-type dynamically later via the API or the website. The configuration that is generated will not overwrite this entity-type.

There are some additional keywords that can be used on Types:
overridable, enum, auto_expand, fuzzy_extract
These each set a setting property in Dialogflow.

Intents

Configuring intents is what makes agents run.

A simple intent might look something like this:

```
    Intent TellJoke
        trained with phrase 'Tell me a Joke', 'Do you know any Jokes', 'Tell me
something funny'
        response "Sure. What is brown and sticky? A Stick!"
```

Now most likely you will want to do something more sophisticated and maybe get some input from the user. In order to use any custom entities in your intents you will first need to configure them as a Type (see above). Built-in entities can be used as is.
A more sophisticated agent might look something like this:

```
Agent WeatherSample
    language en
    Webhook
        active
        url 'https://sample.weatherservice.com/agentresponse'
    Intent CheckWeather
        parameters Place geo_city (required)
        trained with phrase 'What is the weather like in' Place
        webhook_fullfillment
```

There are a large number of additional keywords that can be used in an Intent. Below you can find a list:

Contexts

Contexts can be used to make data available for later intents or to make data required for an intent:

```
Agent SampleAgent
    language en
    version 0.1.1
    description 'This is a sample agent to show how the .dfc language is
specified.'
    Webhook
        active
        url 'https://sample.weatherservice.com/agentresponse'
    Intent CheckWeather
        contexts
            input Place
            output Place, Weather
        trained with phrase 'What is the weather like here'
        webhook_fullfillment
```

Parameters

A parameters can have additional configuration.
syntax:

```
Intent [IntentName]
    parameters [ParameterName] [EntityType] (list required prompts 'Prompt text')
```

Including the list keyword makes the Parameter accept multiple values as a lst.
Including the required keyword makes the parameter required.
The prompts keyword can be given one or more strings to use as prompts if the user did not give a parameter.

Training Phrases

Training phrases can be manually entered if only a small number are used, or a file with preconfigured training phrases can be passed in.

In the case of entering the phrases directly the syntax is exactly as seen above:

```
trained with phrase 'Training Pharse here', 'Another Training Phrase'
```

if a file is passed in the syntax used is:

```
trained with file 'filename.json'
```

The content of this file is not edited in any way by the compiler, and only the filename is changed to match what google expects for a training phrase file for the given intent.

Responses

Responses are very simple. Simply enter the keyword `response` and any number of strings from which the agent will pick it's response.
If rich responses are needed I recommend using the `webhook_fullfillment` keyword.

Aditional Settings

There are some additional settings that can be used. So we have allready seen like the `webhook_fullfillment` keyword. The possible settings are:
`events 'one or more' 'Strings' ...` These will be passed on to your fullfillment or handles by Dialogflow. See https://dialogflow.com/docs/events
`action 'only one String'` This will be passed on to your fullfillment. See https://dialogflow.com/docs/intents/actions-parameters
`webhook_fullfillment` Sets this intent to use your webhook for fullfillment. `webhook_slot_filling` Sets this intent to use your webhook for slot filling. `disable_ml` Turns of ML auto expansion on the training phrases of this intent. `end_of_conversation` Sets this intent to end the conversation.