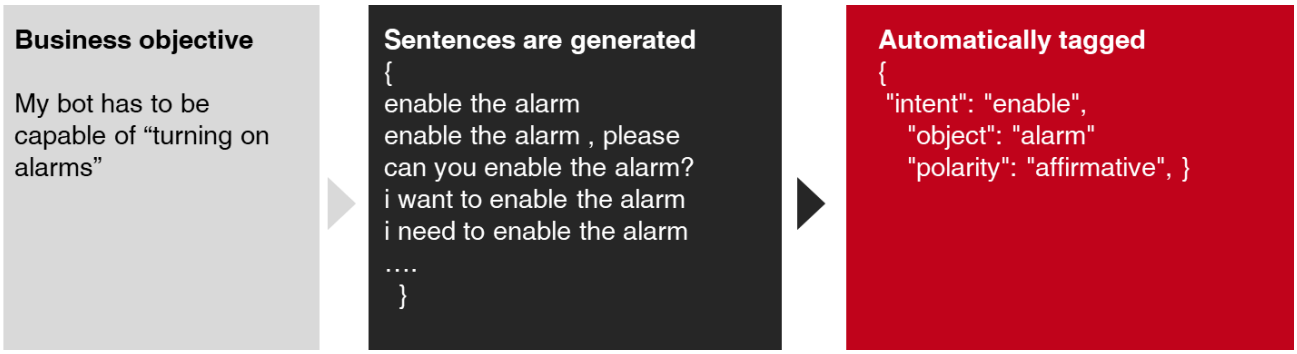
Bitext NLG Integration with Dialogflow

Background on Natural Language Generation

The creation of chatbots requires long development cycles with a lot of man hours and uncertain outcomes. Our NLP Middleware will help you to solve the problem of data sparsity **generating hundreds of relevant queries and automatically tagging** them with the intents and entities you need your bot to recognize.

Key points:

- Training data quantity and quality has a direct impact on your AI performance. **Improvements of over 60%** in bot understanding.
- Automating the data generation allows to develop bots in days, not months.
- Upload the training data to Dialogflow through their API and forget about manual inputs.



Dialogflow integration guide

How to upload training data in three steps using Python and the Dialogflow API

1) Create the agent in Dialogflow
2) Upload entities
3) Upload intents

1) Create the agent in Dialogflow

Go to the Dialogflow console and sign in with your Google account. Click ‘Create agent’ and give your agent a name, that’s enough to continue. Click ‘Save’.

Go to the bot setting by clicking the gear icon button and copy the developer access token. It is necessary to upload both entities and intents.

2) Upload entities

The API request needs an authorization bearer: just paste the developer access token right next to ‘bearer’ instead of the X’s.

Script sample for uploading 2 entities (‘action’, containing the values ‘enable’, ‘turn on’, ‘switch on’ and ‘activate’, and ‘object’, containing the value ‘alarm’):

```
      import json
      import requests
      from time import sleep

      # Define the function to upload entities out of a list
      def apiai_client_put_entities(all_entities):
          for entity in all_entities:
              entities_json = json.dumps(entity)
              res =
requests.put('https://api.api.ai/v1/entities?v=20170712', headers={
"Authorization" : "bearer XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX", "Content-
Type": "application/json; charset=utf-8"}, data=entities_json)
              if not res.status_code == 200:
                  print(entities_json)
              print(res.status_code)
              sleep(1)

      # Define the list of entities
```

```
    entities = [
        {
            "name": "action",
            "entries": [
                {
                    "synonyms": [],
                    "value": "enable"
                },
                {
                    "synonyms": [],
                    "value": "turn on"
                },
                {
                    "synonyms": [],
                    "value": "switch on"
                },
                {
                    "synonyms": [],
                    "value": "activate"
                }
            ]
        },
        {
            "name": "object",
            "entries": [
                {
                    "synonyms": [],
                    "value": "alarm"
                }
            ]
        }
    ]

    # Upload entities
    apiai_client_put_entities(entities)
```

3) Upload intents

Again, the API request needs an authorization bearer: just paste the developer access token right next to ‘bearer’ instead of the X’s.

Script sample for uploading an intent named ‘EnableAlarm’:

```
    import json
    import requests
    from time import sleep

    # Define the function to upload intents out of a list
    def apiai_client_post_intents(all_intents):
        for intent in all_intents:
            intent_json = json.dumps(intent)
            res =
requests.post('https://api.api.ai/v1/intents?v=20170712', headers={
```

```
"Authorization" : "bearer XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX", "Content-
Type": "application/json; charset=utf-8" }, data=intent_json)
            if not res.status_code == 200:
                print(intent_json)
            print(res.status_code)
            sleep(1)

    # Define the list of intents
    intents = [
        {
        "name": "EnableAlarm",
        "auto": True,
        "templates": [
            "@action:enable @object:alarm"
        ],
        "userSays": [],
        "contexts": [],
        "responses": [
            {
                "messages": [
                    {
                        "type": 0,
                        "speech": [
                            "Ok, I will $enable the $alarm!"
                        ],
                        "lang": "en"
                    }
                ],
                "affectedContexts": [],
                "speech": [],
                "action": "EnableAlarm",
                "parameters": [
                    {
                        "prompts": [
                            "what should I $enable?"
                        ],
                        "name": "alarm",
                        "required": True,
                        "dataType": "@object",
                        "value": "$alarm"
                    },
                    {
                        "name": "enable",
                        "required": False,
                        "dataType": "@action",
                        "value": "$enable"
                    }
                ],
                "resetContexts": False
            }
        ],
        "priority": 500000
    }
    ]

    # Upload intents
    apiai_client_post_intents(intents)
```