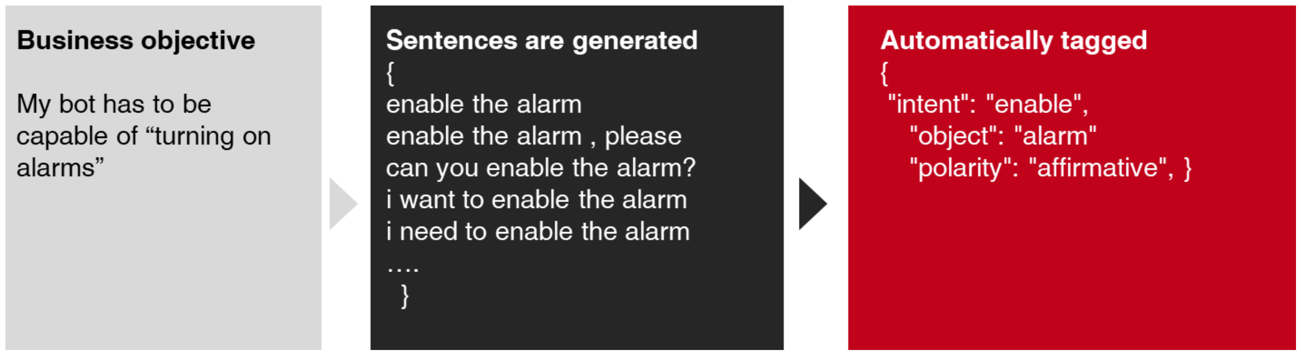
Bitext NLG Integration with LUIS

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 LUIS through their API and forget about manual inputs.



Business objective

My bot has to be capable of “turning on alarms”

Sentences are generated

```
{
enable the alarm
enable the alarm , please
can you enable the alarm?
i want to enable the alarm
i need to enable the alarm
….
 }
```

Automatically tagged

```
{
 "intent": "enable",
   "object": "alarm"
   "polarity": "affirmative", }
```

Microsoft LUIS integration guide

How to upload training data in three steps using SSH and the LUIS API

1) Get LUIS credentials
2) Build the bot JSON
3) Upload the bot via SSH
4) Train the bot

1) Get LUIS credentials

Log in or create an account at Luis.ai and go to your account settings to copy your programmatic key.

2) Build the bot JSON

You can fit the whole bot into a JSON. Note that the entities must be tagged in the utterances, by providing the offset, the beginning and the end position of the entity in the text.

Sample JSON for a bot with one intent, two entities ("action", which has the values "enable", "turn on", "switch on" and "activate", and "object", which has the values "alarm" and "clock"), and 4 utterances:

```
{
"name": "YourBot",
"desc": "",
"culture": "en-us",
"intents": [
    {
        "Name": "EnableAlarm"
    }
],
"entities": [
    {
        "Name": "action"
    },
    {
        "Name": "object"
    }
],
"bing_entities": [],
"actions": [],
"model_features": [],
"regex_features": [],
"utterances": [
    {
        "text": "enable alarm",
        "intent": "EnableAlarm",
        "entities": [
            {
                "entity": "action",
                "_entity_text": "enable",
                "startPos": 0,
                "endPos": 5
            },
            {
                "entity": "object",
                "_entity_text": "alarm",
                "startPos": 7,
                "endPos": 11
            }
        ]
    },
    {
        "text": "turn on alarm",
        "intent": "EnableAlarm",
        "entities": [
            {
                "entity": "action",
                "_entity_text": "turn on",
                "startPos": 0,
                "endPos": 6
            },
            {
                "entity": "object",
                "_entity_text": "alarm",
                "startPos": 8,
                "endPos": 12
            }
        ]
    },
    {
        "text": "switch on clock",
        "intent": "EnableAlarm",
        "entities": [
            {
                "entity": "action",
                "_entity_text": "switch on",
                "startPos": 0,
                "endPos": 8
            },
            {
                "entity": "object",
                "_entity_text": "clock",
                "startPos": 10,
                "endPos": 14
            }
        ]
    },
      {
          "text": "activate clock",
          "intent": "EnableAlarm",
          "entities": [
              {
                  "entity": "action",
                  "_entity_text": "activate",
                  "startPos": 0,
                  "endPos": 7
              },
              {
                  "entity": "object",
                  "_entity_text": "clock",
                  "startPos": 9,
                  "endPos": 13
              }
          ]
      }
  ]
  }
```

3) Upload the bot

You can simply upload the bot via SSH. All you need is:

- The programmatic key: paste it as the value of ‘key’ instead of the X’s
- The name you want to give to your LUIS application: paste it as the value of ‘appName’ inside the URL, instead of “YourApp”
- The name of your JSON file containing the bot data: paste it at the end, right next to the ‘@’, instead of ‘Bot.json’

Sample script:

```
      key="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

      xid0=`curl -k -X POST -H "Content-Type: application/json" -H
"Ocp-Apim-Subscription-Key: ${key}"
"https://westus.api.cognitive.microsoft.com/luis/v1.0/prog/apps/import
?appName=YourBot" -d@Bot.json`
```

4) Train the bot

Before the testing of your bot, you have to train it. LUIS training is faster via web than programmatically: simply go to your Applications page, click on the name of the bot you have just created, and then ‘Train & Test’. Finally press the ‘Train’ button and the app will be ready to be tested.