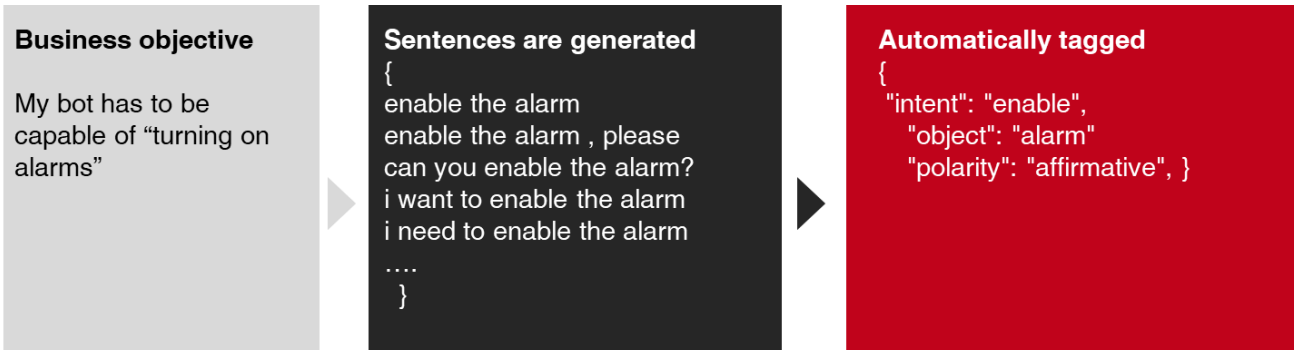
Bitext NLG Integration with Lex

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 Lex through its SDK and forget about manual inputs.



Amazon Lex integration guide

How to upload training data in six steps using the Amazon Lex SDK

1) AWS Credentials
2) Create a bot in AWS
3) Create a file with AWS user credentials
4) Install the SDK
5) Create script to send training data to the bot
6) Use the SDK to generate the bot through the AWS API

1) AWS Credentials

You need these to authenticate later, in the programmatic access to Lex.

a. Create an IAM user in our AWS account: https://console.aws.amazon.com/iam/home#/home. First, you have to create a group with AmazonLexFullAccess permissions, then create a user and add it to the group.
b. Generate an authentication key for that user: go to the user details, click on 'Security credentials', then 'Create Access key', and copy the **'aws_access_key_id'** and the **'aws_secret_access_key'** that will generate. Save them as you will need them later.

2) Create a bot in AWS

a. Log in the AWS console, and go to https://console.aws.amazon.com/lex/home?region=us-east-1. Create a new bot and save its **name**, because you are going to use it to authenticate.
b. In the console you can create intents, slot types and prompts without any line of code.

   The conversational logic is easy: each intent has a series of slot types assigned. When the bot finds all the slot types in the user input, it

delivers the confirmation prompt. If any slot type is missing, it delivers the elicitation prompt associated to that slot type.

c. Once the bot is configured, you will have to click 'Build' and then 'Publish'. To do that you have to set an **alias**, which will also be used in the authentication during the programmatic access.

3) Create a file with AWS user credentials

(https://boto3.readthedocs.io/en/latest/guide/quickstart.html#configuration).

It has to be located in the ~/.aws/config directory and contain the following lines (changing the key values to the ones you saved earlier):

```
[default]
aws_access_key_id = XXXXXXXXXXXXXXXXXXX
aws_secret_access_key = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX/XXXXXX
region = us-east-1
```

4) Install the SDK

```
> pip3 install boto3
```

5) Create a script to send training data to the bot

(https://boto3.readthedocs.io/en/latest/reference/services/lex-runtime.html)

Sample script:

```
import boto3
client = boto3.client('lex-runtime')

response = client.post_text(
    botName='BookTrip',
    botAlias='dev',
    userId='string',
    sessionAttributes={'string': 'string'},
    inputText='I want a car'
)

print(response)
```

6) Use the SDK to generate the Bot trough the AWS API

(https://boto3.readthedocs.io/en/latest/reference/services/lex-models.html)

6.1) First, you have to upload the slot types. You can upload all of them at once in a single script, adding the necessary 'response' variables.

For example, a script to upload a slot type named 'enable', containing the slot values "enable", "turn on", "switch on" and "activate":

```
import boto3
client = boto3.client('lex-models')

response = client.put_slot_type(
        name='enable',
        description='',
        enumerationValues=[
        {
            "value": "enable",
            "synonyms": [
                "enable"
            ]
        },
        {
            "value": "turn on",
            "synonyms": [
                "turn on"
            ]
        },
        {
            "value": "switch on",
            "synonyms": [
                "switch on"
            ]
        },
        {
            "value": "activate",
            "synonyms": [
                "activate"
            ]
        }
    ],
#    checksum='1',
#    valueSelectionStrategy='ORIGINAL_VALUE'|'TOP_RESOLUTION'
)
```

6.2) Then you can upload the intents. Note that there's a limit of 5 confirmation prompts. You can also upload all the intents at once the same way you uploaded the slot types.

For example, a script to upload an intent named ‘EnableAlarm’:

```
      import boto3
      client = boto3.client("lex-models")

      response = client.put_intent(
          name="EnableAlarm",
          description="",
          slots=[
              {
                  "name": "alarm",
                  "slotConstraint": "Required",
                  "slotType": "alarm",
                  "slotTypeVersion": "$LATEST",
                  "valueElicitationPrompt": {
                      "messages": [
                          {
                              "contentType": "PlainText",
                              "content": "What should I {enable}?"
                          }, {
                              "contentType": "PlainText",
                              "content": "What do you want me to
{enable}?"
                          }
                      ],
                      "maxAttempts": 3,
                      "responseCard": "string"
                  },
                  "priority": 3,
                  "sampleUtterances": [],
                  "responseCard": "string"
              }, {
                  "name": "enable",
                  "slotConstraint": "Required",
                  "slotType": "enable",
                  "slotTypeVersion": "$LATEST",
                  "valueElicitationPrompt": {
                      "messages": [
                          {
                              "contentType": "PlainText",
                              "content": "What should I do with the
{alarm}?"
                          }
                      ],
                      "maxAttempts": 3,
                      "responseCard": "string"
                  },
                  "priority": 3,
                  "sampleUtterances": [],
                  "responseCard": "string"
              }
          ],
          confirmationPrompt={
```

```
            "messages": [
                {
                    "contentType": "PlainText",
                    "content": "I will {enable} the {alarm}!"
                }
            ],
            "maxAttempts": 3,
            "responseCard": "string"
        },
        rejectionStatement={
            "messages": [
                {
                    "contentType": "PlainText",
                    "content": "Ok, I won't {enable} anything"
                }
            ],
            "responseCard": "string"
        },
        sampleUtterances=[
            "{enable} {alarm}"
            ],
    #    parentIntentSignature="0001",
    #    checksum = ""
    )
```