



Technical Note REST API Description

REST API description. Version 1.1. Jul 2017.



Document's information

Project	BITEXT LEMMATIZATION API
Description	REST API description. Version 1.1. Jul 2017.
Author (s)	Luis Miguel Serrano; Luis Caloto
Customer / Dept.	
Last Mod. Date	July 12 th , 2017
Issue	A
DOC_ID	

Table of contents

1	Purpose	4
2	Scope	4
3	Documents	4
3.1	Applicable Documents	4
3.2	Reference Documents	4
4	Definitions and Acronyms	5
4.1	Definitions	5
4.2	Acronyms and Abbreviations	5
6	REST API	¡Error! Marcador no definido.
7	Lemmatization Analysis	8
7.1	POST Request	8
7.2	GET Request	9
7.3	Example	10
8	Code Samples	11
8.1	Perl	11
8.2	Python	13
8.3	Ruby	15
8.4	Java	17

1 Purpose

This document describes how to use the Bitext Lemmatization API.

2 Scope

This document applies to the integration and adaptation of the Bitext Lemmatization API, provided by Bitext.

Descriptions and instructions contained in this document might be modified in future versions of this same document.

3 Documents

3.1 Applicable Documents

Code	Name	Version
N/A		

--	--	--

3.2 Reference Documents

Code	Name	Version
N/A		

4 Definitions and Acronyms

4.1 Definitions

4.2 Acronyms and Abbreviations

ASCII	<i>American Standard Code for Information Interchange</i> (character encoding set)
API	<i>Application Programming Interface</i>
SW	<i>Software</i>
CSV	<i>Comma Separated Values</i> – (structured plain text format)
UTF-8	<i>UCS Transformation Format – 8 bits</i> (character encoding set)
UCS	<i>Universal Character Set</i> (character set standard)
ISO-8859-1	Character encoding set based on Latin character set for most Western European languages used mainly in Unix/Linux Operating Systems
Windows1252	Character encoding set based on Latin character set for most Western European languages used mainly in Windows Operating Systems

6 REST API

The Bitext Lemmatization API can be reached by using the following endpoint:

`https://svc02.api.bitext.com/lemmatization/`

For this API, the following languages are available: Standard Arabic (MSA), English, French, German, Italian, Korean, Dutch, Norwegian Bokmål, Norwegian Nynorsk, Persian (Farsi), Portuguese, Russian, Spanish and Swedish.

You must specify the language code as parameter for requests:

Language	Code
Standard Arabic (MSA)	arb
Dutch	nld
English	eng
French	fra
German	deu
Italian	ita
Korean	kor
Norwegian Bokmål	nob
Norwegian Nynorsk	nno
Persian (Farsi)	fas
Portuguese	por
Russian	rus
Spanish	spa
Swedish	swe

A person identification token should be added to all requests. Such a token is provided by Bitext after creating an account in the API website, below My Profile > API Credentials. For this document we'll use the example token "tokenprovidedbybitext".

The token must be sent as the value of the Authorization header, and should appear with a bearer. For example, using "curl", you can reach the Bitext API like this:

```
curl -X POST --url https://svc02.api.bitext.com/lemmatization/ -H "Authorization : bearer tokenprovidedbybitext"
```

If you expect a JSON response, you must also specify so in the headers. You must instantiate the header “Content-Type” with the “application/json” value.

```
curl -X POST --url https://svc02.api.bitext.com/lemmatization/ -H "Authorization : bearer tokenprovidedbybitext" -H "Content-Type: application/json"
```

The Bitext API works asynchronously. You must first request the API to start analyzing a text. After this first request, you can perform additional requests, until the API eventually returns the correct analysis for the text included in the first request.

The first request is a POST request that sends a JSON with two parameters: language and text. The value of the text parameter corresponds to the list of words to be analyzed, and the value of the language parameter corresponds to the language of those words.

Currently, the limit of characters per API call is 50.000

For example, in “curl”, a first request for the analysis of “lost” and “can’t” will look as follows:

```
curl -X POST --url https://svc02.api.bitext.com/lemmatization/ -H "Authorization : bearer tokenprovidedbybitext" -H "Content-Type: application/json" --data '{"language":"eng", "text":["lost", "can\t"]}'
```

This request is calling to the “lemmatization analysis” endpoint. The API will immediately return a JSON response that looks like this:

```
{
  "resultid": "9a23759db4ae46d8bfbbab47ac4dff1f",
  "success": true,
  "message": "Request accepted"
}
```

The **resultid** field in this response is crucial. It will be used to build the subsequent requests asking if the result for a given previous request is already available or not. These new requests are GET type (not POST) and they MUST include the **resultid** in the URL. To ask the Bitext API if the results for one of the texts previously sent, you would write the following request with “curl”:

```
curl -X GET --url https://svc02.api.bitext.com/lemmatization/9a23759db4ae46d8bfbbab47ac4dff1f/ -H "Authorization: bearer tokenprovidedbybitext" -H "Content-Type : application/json"
```

If the analysis is already available, the response to that request would be the following:

```
{
  "resultid": "9a23759db4ae46d8bfbbab47ac4dff1f",
  "lemmatizationanalysis": [
    {
      "form": "can't",
      "lemmas": ["can not"]
    },
    {
      "form": "lost",
      "lemmas": ["lose", "lost"]
    }
  ]
}
```

If the analysis is not yet available the (202 HTTP code) will be returned, and you will need to try the same request again.

7 Lemmatization Analysis

The Bitext Lemmatization service provides all potential roots (also known as lemmas) for a given word, using morphological analysis and carefully curated lexicons. For example, for the word "spoke", the lemmatization service will return the lemmas "speak" and "spoke". For the sentence "We spoke yesterday about fixing the bike.", the lemmatization service will output a set of lemmas for every word in the sentence.

The lemmatization analysis API endpoint is supposed to receive a list of words. For every word in that input list, the service retrieves a list of lemmas. If no lemmas have been found for a word, it will have an empty list.

7.1 POST Request

Request

- Endpoint: /lemmatization/
- Protocol: HTTPS
- URL: <https://svc02.api.bitext.com/lemmatization/>
- Method: POST
- Headers:

```
{
  "Authorization": "bearer tokenprovidedbybitext",
  "Content-Type": "application/json"
}
```

- Data parameters:


```
{
  "language": "eng",
  "text": ["...", "..."]
}
```

Being "**text**", the list of words you want to analyze. One or more, in double quotes and separated by commas.

Currently, the limit of word per lemmatization API call is 100.

Successful responses:

- Code: 201

```
{
  "success": true,
  "message": "Request accepted",
  "resultid": "...
}
```

7.2 GET Request

Request

- Endpoint: /lemmatization/
- Protocol: HTTPS
- URL: <https://svc02.api.bitext.com/lemmatization/:resultid/>
- Method: GET
- Headers:

```
{
  "Authorization": "bearer tokenprovidedbybitext",
  "Content-Type": "application/json"
}
```

Being "**resultid**" the identifier of the analysis request (retrieved by the POST transaction)

Successful responses:

- Code: 202

The analysis with ID "**resultid**" is not yet complete and must be requested again.

```
{
  "resultid": "...
}
```

- Code: 200

```
{
  "success": true,
  "resultid": "...",
  "lemmatizationanalysis": [
    {
      "form": "...",
      "lemmas": ["...", "..."]
    },
    {
      "form": "...",
      "lemmas": ["...", "..."]
    },
    {
      "form": "...",
      "lemmas": ["...", "..."]
    }
  ]
}
```

- Code 200: The analysis with ID ***resultid*** is complete
- resultid: Identifier of the analysis request (retrieved by the POST transaction and sent in the current request)
- form: Analyzed word
- lemmas: List of lemmas found for the word analyzed

7.3 Example

POST Request

```
-s -X POST --url https://svc02.api.bitext.com/lemmatization/ -H "Authorization: bearer tokenprovidedbybitext" -H "Content-Type: application/json" -- data '{"language":"eng", "text":["lost", "can't"]}'
```

POST Response

```
{
  "success": true,
  "message": "Request accepted",
  "resultid": "123451234512345123451234512345ab"
}
```

GET Request

```
curl -s -X GET --url https://svc02.api.bitext.com/lemmatization/123456789abcdefg123456789abcdefg/ -H "Authorization: bearer tokenprovidedbybitext" -H "Content-Type: application/json"
```

GET Response

```
{
  "resultid": "1b84e78a8655448db6b1730927fba1a2",
  "lemmatizationanalysis": [
    {
      "form": "can't",
      "lemmas": ["can not"]
    },
    {
      "form": "lost",
      "lemmas": ["lose", "lost"]
    }
  ]
}
```

8 Code Samples

8.1 Perl

```
use Mojo::UserAgent;
use Mojo::JSON qw(encode_json);

# User token, required for API access
my $oauth_token = "tokenprovidedbybitext";

# API request data: Language and text to be analyzed
my $user_language = "eng";
my $user_text = ["I", "have", "a", "beautiful", "house", "in", "Madrid"];

print "\nBITEXT API. Lemmatization endpoint. PERL sample code\n";
print "-----\n\n";

my $ua = Mojo::UserAgent->new;

# Building the POST request to lemmatization analysis endpoint
my $endpoint = "https://svc02.api.bitext.com/lemmatization/";
my $headers = { Authorization => "bearer $oauth_token", 'Content-Type' => 'application/json' };
my $params = {"language" => "$user_language", "text" => "$user_text"};

# Sending the POST request
$tx = $ua->post($endpoint, $headers, json => $params);

# Processing the result of the POST request
```

```

my $post_result = $tx->res->json->{success};           # Success of the request
my $post_result_code = $tx->res->code;                 # Error code, if applicable
my $post_msg = $tx->res->json->{message};             # Error message, if applicable
my $action_id = $tx->res->json->{resultid};           # Identifier to request the analysis results

print "POST: '$post_msg'\n\n";

# 401 is the error code corresponding to an invalid token
if ($post_result_code == 401)
{
    print "Your authentication token is not correct\n";
}

if ($post_result)
{
    print "Waiting for analysis results...\n\n";

    # GET request loop, using the response identifier returned in the POST
    answer
    my $analysis;
    until ($analysis)
    {
        $tx = $ua->get($endpoint.$action_id.'/', $headers);
        eval { $analysis = $tx->res->json };
    }

    # The loop ends when we have response to the GET request
    my $get_msg = $tx->res->message;
    print "GET: '$get_msg'\n\n";

    # In the GET response we have the result of the analysis
    print "Analysis results:\n\n";
    print encode_json $analysis;
    print "\n";
}

```

8.2 Python

```
import requests, json;

# User token, required for API access
oauth_token = 'tokenprovidedbybitext'

# API request data: Language and text to be analyzed
user_language = "eng"
user_text = ["I", "have", "a", "beautiful", "house", "in", "Madrid"]

print ("\nBITEXT API. Lemmatization endpoint. PYTHON sample code\n")
print ("-----\n\n")

# Building the POST request to lemmatization analysis endpoint
endpoint = "https://svc02.api.bitext.com/lemmatization/"
headers = { "Authorization" : "bearer " + oauth_token, "Content-Type" : "application/json" }
params = { "language" : user_language, "text" : user_text }

# Sending the POST request
res = requests.post ( endpoint, headers=headers, data=json.dumps(params) )

# Processing the result of the POST request
post_result = json.loads(res.text).get('success')           # Success of the request
post_result_code = res.status_code                         # Error code, if applicable
post_msg = json.loads(res.text).get('message')            # Error message, if applicable
action_id = json.loads(res.text).get('resultid')          # Identifier to request the analysis results

print ( "POST: '" + post_msg + "'\n\n" )

# 401 is the error code corresponding to an invalid token
if ( post_result_code == 401 ):
    print ( "Your authentication token is not correct\n" )

if ( post_result ):
    print ( "Waiting for analysis results...\n\n" )

    # GET request loop, using the response identifier returned in the POST answer
    analysis = None
    while analysis == None:
        res = requests.get(endpoint + action_id + '/', headers=headers)

        if res.status_code == 200 :
            analysis = res.text
```

```
# The loop ends when we have response to the GET request
get_msg = res.reason
print ( "GET: '" + get_msg + "'\n\n" )

# In the GET response we have the result of the analysis
print ( "Analysis results:\n\n" )
print ( analysis )
print ( "\n" )
```

8.3 Ruby

```
require 'httpclient'
require 'json'

# User token, required for API access
oauth_token = 'tokenprovidedbybitext'

# API request data: Language and text to be analyzed
user_language = 'eng'
user_text = ["I", "have", "a", "beautiful", "house", "in", "Madrid"]

puts ''
puts 'BITEXT API. Lemmatization endpoint. Ruby sample code'
puts '-----'
puts ''

clnt = HTTPClient.new

# Building the POST request to lemmatization analysis endpoint
endpoint = "https://svc02.api.bitext.com/lemmatization/"
headers = { "Authorization" => "bearer " + oauth_token, "Content-Type" => "application/json"}
params = {"language" => "" + user_language + "" , "text" => "" + user_text + ""}
.to_json

# Sending the POST request
res = clnt.post(endpoint,params,headers)

# Processing the result of the POST request
content = JSON.parse(res.content)

post_result_code = res.status # Error code, if applicable
action_id = content["resultid"] # Identifier to request the analysis results
post_msg = content["message"] # Error message, if applicable
post_result = content["success"] # Success of the request

puts 'POST: ' + post_msg
puts ''

# 401 is the error code corresponding to an invalid token
if ( post_result_code == 401 )
  puts 'Your authentication token is not correct'
end
```

```

if ( post_result )

    puts 'Waiting for analysis results...'
    puts ''

    # GET request loop, using the response identifier returned in the POST
answer
    analysis = ""

    while analysis == "" do

        res = clnt.get(endpoint + action_id + "/", {}, headers)
        if res.status == 200
            analysis = res.content
        elsif res.status != 202
            puts "GET: Error code (#{res.status}), message: " +
res.reason
                exit
        end
    end

    # The loop ends when we have response to the GET request
    get_msg = res.reason;
    puts 'GET: ' + get_msg
    puts ''

    # In the GET response we have the result of the analysis
    puts 'Analysis results:'
    puts ''
    puts analysis
    puts ''
end

```


8.4 Java

```
import java.net.HttpURLConnection;
import java.net.URL;
import java.io.*;
import org.json.JSONObject;

public class codeSample {

    // User token, required for API access
    public static String oauth_token = "tokenprovidedbybitext";

    // API request data: Language and text to be analyzed
    public static String user_language = "eng";
    public static List user_text = ["I", "have", "a", "beautiful", "house"
, "in", "Madrid"];

    // Building the POST request to lemmatization analysis endpoint
    public static String endpoint = "https://svc02.api.bitext.com/lemmatization/";
    public static String textdata = "{\"language\":\"" + user_language + "\",\"text\":\"" + user_text + "\"}";

    public static void main(String[] args) throws Exception
    {
        System.out.println( "\nBITEXT API. Lemmatization endpoint. JAVA sample code" );
        System.out.println( "-----\n" );

        // Sending the POST request
        URL urlPOST = new URL(endpoint);
        HttpURLConnection connectionPOST = createAPIConnection("POST",urlPOST);
        DataOutputStream outputStream = new DataOutputStream(connectionPOST.getOutputStream());
        outputStream.write(textdata.getBytes("UTF8"));
        outputStream.flush();
        outputStream.close();

        // Processing the response code of the POST request
        switch (connectionPOST.getResponseCode())
        {
            case 201: // 201 is the code for succesful request processing
                break;
            case 401: // 401 is the error code corresponding to an invalid token
                System.out.println ( "Your authentication token is not correct" );
                System.exit(0);
            case 402: System.out.println ( "No contract found for that language" );
        }
    }
}
```

```

        System.exit(0);
        default:
            break;
    }

    // Processing the result of the POST request
    String sResponse1 = getAPIResponse(connectionPOST);
    JSONObject jsonResponse1 = new JSONObject(sResponse1);
    String action_id = jsonResponse1.getString("resultid"); // Identifier to request the analysis results
    String post_msg = jsonResponse1.getString("message");
    // Error message, if applicable
    boolean post_result = jsonResponse1.getBoolean("success");
    // Success of the request

    System.out.println ( "POST: " + post_msg + "\n");

    if (post_result)
    {
        System.out.println ( "Waiting for analysis results..
.\n" );

        // GET request loop, using the response identifier returned in the POST answer
        // Ask for the result of the analysis launched before
        // Try until the analysis is ready and the API returns it

        URL urlGET = new URL(endpoint + action_id + "/");
        String sResponse2 = "";
        HttpURLConnection connectionGET = null;

        while (sResponse2 == "")
        {
            connectionGET = createAPIConnection("GET",urlGET);
            if (connectionGET.getResponseCode() == 200)
            {
                sResponse2 = getAPIResponse(connectionGET);
            }
        }

        // The loop ends when we have response to the GET request
        System.out.println ( "POST: " + connectionGET.getResponseMessage() + "\n");

        // In the GET response we have the result of the analysis
        System.out.println ( "Analysis results:\n" );
    }
}

```

```

        System.out.println(sResponse2); // Print it with spe
specified indentation
        System.out.println ( "" );
    }
}

    public static HttpURLConnection createAPIConnection(String method, URL
url) throws Exception
    {

        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod(method);
        conn.setRequestProperty("Content-Type", "application/json");
        conn.setRequestProperty("Authorization", "bearer " + oauth_token);

        if (method == "POST")
        {
            conn.setDoOutput(true);
            conn.setDoInput(true);
        }
        return conn;
    }

    public static String getAPIResponse(HttpURLConnection conn) throws Exception
    {

        String sResponse = "";
        BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInp
utStream()));
        String decodedString;

        while ((decodedString = in.readLine()) != null)
        {
            sResponse = sResponse+"\n"+decodedString;
        }
        sResponse = sResponse.trim();
        in.close();
        return sResponse;
    }
}

```