Bitext API. Quick Start Guide

1. Bitext Control Panel

Everything you need to work with Bitext, including text analysis tools, account management, access to Bitext products and services, support, and an interactive API testing tool – which lets you test the Bitext API – are all accessible right from in the Bitext control panel.

Each time you log in to Bitext, you arrive at the main page of the Bitext control panel. There are four tabs on the left-hand side of the main page: My Profile, My Subscriptions, Add Subscriptions, API Test Tool and Quick Start Guide.

1.1 My Profile

Your account information is accessible from the My Profile tab on the left-hand side of the control panel.

To review or update your personal information, you can click Personal Info tab. To change your password, you can click Change Password tab

The API Credentials tab is particularly important because it is here where you could will find your user authentication token that you will need to perform any request using Bitext API from your own code.

The My Profile screen is also accessible from the top right-hand corner of the control panel, where your email appears. When you click your email, a dropdown menu gives you two options:

- **My Profile:**

  View and edit your account information

- **CX Mode / API Mode:**

  View and edit your account information

- **Log Out:**

  Close your control panel session

2. API REST

The Bitext API can be reached by using the following 4 endpoints:

- https://svc02.api.bitext.com/sentiment/
- https://svc02.api.bitext.com/entities/
- https://svc02.api.bitext.com/concepts/
- https://svc02.api.bitext.com/categories/

The endpoints correspond to the subscriptions you have activated on the My Subscriptions page. For example, if you want to perform a sentiment analysis in English, you must first have an active free trial or paid subscription for sentiment analysis in English activated in your account. You would then use the endpoint https://svc02.api.bitext.com/sentiment and specify English as the language parameter for requests.

In order to make the endpoint work, you must connect your credentials to the endpoint, by adding your personal authentication token to all requests. Your authentication token can be found in the My Profile section of the dashboard. Click API Credentials, copy the token and use it in your code. The token must be sent as the value of the Authorization header, and should appear with a bearer.

For example, if your personal token is "abcdefghijklmnopqrstuvwxyz" then, using Curl, you can reach the Bitext API like this:

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

You must also specify in the headers that you expect a JSON response, and must then instantiate the header Content-Type with the application/json value.

```
curl -X POST --url https://svc02.api.bitext.com/sentiment/ -H "Authorization: bearer abcdefghijklmnopqrstvwxyz" -H "C
ontent-Type: application/json"
```

The Bitext API works asynchronously. You must first request that the API start analyzing a text, after which you can perform additional requests, until the API returns the correct analysis of the text.

The first request is a POST request that sends a JSON with two parameters: language and text. The value of the text parameter corresponds with the text to be analyzed, and the value of the language parameter corresponds with the language of that text. Currently the limit of characters per API call is of 1000. We are working to increase that limit and we will do so promptly. For example, in Curl, a first request for the analysis of "I have a beautiful house in Madrid" would appear as follows:

```
curl -X POST --url https://svc02.api.bitext.com/sentiment/ -H "Authorization: bearer abcdefghijklmnopqrstvwxyz" -H "C
ontent-Type: application/json" --data '{"language":"eng", "text":"I have a beautiful house in Madrid"}'
```

This request is calling to the endpoint sentiment analysis in English (the parameter language). The action will only be successfully executed if the user currently has an active trial or paid subscription to the sentiment analysis service in English.

To call the "categories" endpoint, you must provide a "codingplanid" parameter. This parameter identifies the coding plan that will be used in order to perform the categorization. The value of the parameter is numeric, and corresponds to the "key" found in your list of coding plans in the Coding Plans screen of the control panel (see: Coding Plans).

```
curl -X POST --url https://svc02.api.bitext.com/categories/ -H "Authorization : bearer abcdefghijklmnopqrstvwxyz", -H "Content-Type : application/json" --data '{"text":"I have a beautiful house in Madrid",”codingplanid”:”123”}'
```

The codes for the currently available Bitext languages are as follows:

- **"eng":** English
- **"cat":** Catalan
- **"deu":** German
- **"fra":** French
- **"ita":** Italian
- **"nld":** Dutch
- **"por":** Portuguese
- **"spa":** Spanish

This first request would return a JSON 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. These new requests are GET and include the resultid in the URL. To ask the Bitext API to further analyze one of the texts previously sent, you would write the following request with Curl:

```
curl -X GET --url https://svc02.api.bitext.com/sentiment/9a23759db4ae46d8bfbbab47ac4dff1f/ -H "Authorization: bear er abcdefghijklmnopqrstvwxyz" -H "Content-Type: application/json"
```

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

```
{
  "resultid": "9a23759db4ae46d8bfbbab47ac4dff1f",
  "sentimentanalysis": [
    {
      "sentence": "I have a beautiful house in Madrid",
      "text_norm": "beautiful",
      "text": "beautiful",
```

```
      "score": "3.000000",
      "topic": "house",
      "topic_norm": "house"
    }
  ]
}
```


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

2.1 Sentiment Analysis

The Bitext sentiment analysis tool analyzes opinions found in texts. It first identifies the topic(s) that are being discussed in a particular text and then evaluates the opinion(s) expressed about the topic(s). The result is a numeric score that is either positive or negative.

For example, for the sentence “I have a beautiful house in Madrid,” Bitext sentiment analysis would identify the topic being discussed as ‘house’ and then establish a relationship between 'house' and 'beautiful’ based on the Bitext engine’s deep knowledge of language. ‘Beautiful' is understood as a positive opinion expressed about the topic ‘house’ and so the numeric score of the sentiment analysis would be positive.

When a text is sent to the sentiment analysis API endpoint, it is divided into different sentences, and analyses are automatically performed on each sentence. If a sentence contains no sentiment expression, it receives a score of ’0’ and the corresponding fields for ‘topics’ and ‘sentiment expressions’ are left blank.

- **POST Request:**

  » Request:

  - Endpoint: /sentiment/
  - URL: https://svc02.api.bitext.com/sentiment/
  - Method: POST
  - Headers:

```
{
  "Authorization": "bearer [token]",
  "Content-Type": "application/json"
}
```

  - Data parameters:

```
{
  "language": "...",
  "text": "..."
}
```

- token: Your API credentials (can be copied from the My Profile section of the control panel)

- language: Language of the text you want to analyze (see available languages below)

- text: Text you want to analyze

Available languages:

- **"eng":** English
- **"cat":** Catalan
- **"deu":** German
- **"fra":** French
- **"ita":** Italian
- **"nld":** Dutch
- **"por":** Portuguese
- **"spa":** Spanish

» Successful responses:

- Code: 201

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

- **GET Request:**

» Request:

- Endpoint: /sentiment/
- URL: https://svc02.api.bitext.com/sentiment/:resultid/
- Method: GET
- Headers:

```
{
  "Authorization": "bearer [token]",
  "Content-Type": "application/json"
}
```

-token: Your API credentials (can be copied from the My Profile section of the control panel)

-resultid: Identifier of the analysis request (retrieved by the POST transaction)

» Successful responses:

- Code: 202

The analysis with ID 'resultid' is not complete and must be called again.


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


- Code: 200


```
{
  "success": true,
  "resultid": "...",
  "sentimentanalysis": [
    {
      "topic": "...",
      "topic_norm": "...",
      "text": "...",
      "text_norm": "...",
      "score": "...",
      "sentence": "..."
    }
  ]
}
```

-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)

-topic: Target of an opinion

-topic_norm: Normalized version of the topic

-text: Opinion about the topic

-text_norm: Normalized version of the text

-sentence: Sentence in which a topic/text relation was found

- **Example:**

» POST Request:

```
-s -X POST --url https://svc02.api.bitext.com/sentiment/ -H "Authorization: be
arer 12345123451234512345123451234512345ab" -H "Content-Type: application/j
son" --data '{"language":"eng", "text":"I have a beautiful house in Madrid"}'
```

» POST Response:


```
{
  "success": true,
  "message": "Request accepted",
  "resultid": "12345123451234512345123451234512345ab"
}
```


» GET Response:

```
curl -s -X GET --url https://svc02.api.bitext.com/sentiment/123456789abcdefg
123456789abcdefg/  -H "Authorization: bearer 12345123451234512345123451234
512345ab" -H "Content-Type: application/json
```

» POST Response:


```
{
  "resultid": "1b84e78a8655448db6b1730927fba1a2",
  "sentimentanalysis": [
    {
      "topic": "house",
      "topic_norm": "house",
      "text": "beautiful",
      "text_norm": "beautiful",
      "score": "3.00000",
      "sentence": "I have a beautiful house in Madrid"
    }
  ]
}
```

2.2 Concept Extraction

The Bitext concept extraction service identifies different kinds of phrases in texts, including nominal phrases such as 'room service' and 'technical assistance,' verbal phrases such as 'like' and 'prefer,' adjectival phrases such as 'great' or 'useful' and adverbial phrases, such as 'successfully' and 'unfortunately.'

When a text is sent to the concepts endpoint, a single bag of concepts is retrieved. Each concept found in the text appears both exactly as it does in the text and in a normalized version (usually the lemma of the word). The type of concept is specified with a numeric code.

- **POST Request:**

  » Request:

  - Endpoint: /concepts/
  - URL: https://svc02.api.bitext.com/concepts/
  - Method: POST
  - Headers:

```
{
  "Authorization": "bearer [token]",
  "Content-Type": "application/json"
}
```

  - Data parameters:

```
{
  "language": "...",
  "text": "..."
}
```

-token: Your API credentials (can be copied from the My Profile section of the control panel)

-language: Language of the text you want to analyze (see available languages below)

-text: Text you want to analyze

Available languages:

- **"eng":** English
- **"cat":** Catalan
- **"deu":** German
- **"fra":** French

- **"ita":** Italian
- **"nld":** Dutch
- **"por":** Portuguese
- **"spa":** Spanish

» Successful responses:

- Code: 201

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

- **GET Request:**

» Request:

- Endpoint: /concepts/
- URL: https://svc02.api.bitext.com/concepts/:resultid/
- Method: GET
- Headers:

```
{
  "Authorization": "bearer [token]",
  "Content-Type": "application/json"
}
```

-token: Your API credentials (can be copied from the My Profile section of the control panel)

-resultid: Identifier of the analysis request (retrieved by the POST transaction)

» Successful responses:

- Code: 202

The analysis with ID 'resultid' is not complete and must be called again.

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

```
}
```

- Code: 200

```
{
  "conceptsanalysis": [
    {
      "type": "...",
      "concept": "...",
      "concept_norm": "..."
    }
  ],
  "resultid": "..."
}
```

-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)

-type: Type of concept identified (see concept types below)

-concept: Concept found (a concept is a nominal, verbal, adverbial or adjectival phrase)

-concept_norm: Normalized version of the concept

Available concepts types:

- **1001:** nominal
- **1002:** adjectival
- **1003:** adverbial
- **1004:** verbal

- **Example:**

» POST Request:

```
-s -X POST --url https://svc02.api.bitext.com/concepts/ -H "Authorization: bearer 12345123451234512345123451234512345ab" -H "Content-Type: application/json" --data '{"language":"eng", "text":"I have a beautiful house in Madrid"}'
```

» POST Response:

```
{
  "success": true,
  "message": "Request accepted",
  "resultid": "12345123451234512345123451234512345ab"
}
```


» GET Response:

curl -s -X GET --url https://svc02.api.bitext.com/concepts/123456789abcdefg123456789abcdefg/ -H "Authorization: bearer 12345123451234512345123451234512345ab" -H "Content-Type: application/json

» POST Response:


```
{
  "conceptsanalysis": [
    {
      "type": "1004",
      "concept": "have",
      "concept_norm": "have"
    },
    {
      "type": "1001",
      "concept": "beautiful house",
      "concept_norm": "beautiful house"
    },
    {
      "type": "1001",
      "concept": "Madrid",
      "concept_norm": "Madrid"
    }
  ],
  "resultid": "91e950cfe0454043b2cb51cd8b7650a4"
}
```

2.3 Entities Extraction

The Bitext entity extraction service finds entities within texts, such as people's names, country names and company names. When a text is sent to the entities endpoint, a single bag of entities is retrieved. Each entity found in the text appears both exactly as it does in the text and also as a normalized version. The type of entity is specified with a numeric code.

- **POST Request:**

  » Request:

  - Endpoint: /entities/
  - URL: https://svc02.api.bitext.com/entities/
  - Method: POST
  - Headers:

```
{
  "Authorization": "bearer [token]",
  "Content-Type": "application/json"
}
```

  - Data parameters:

```
{
  "language": "...",
  "text": "..."
}
```

-token: Your API credentials (can be copied from the My Profile section of the control panel)

-language: Language of the text you want to analyze (see available languages below)

-text: Text you want to analyze

Available languages:

- **"eng":** English
- **"deu":** German
- **"fra":** French
- **"ita":** Italian
- **"nld":** Dutch
- **"por":** Portuguese
- **"spa":** Spanish

» Successful responses:

- Code: 201

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

- **GET Request:**

» Request:

- Endpoint: /entities/
- URL: https://svc02.api.bitext.com/entities/:resultid/
- Method: GET
- Headers:

```
{
  "Authorization": "bearer [token]",
  "Content-Type": "application/json"
}
```

-token: Your API credentials (can be copied from the My Profile section of the control panel)

-resultid: Identifier of the analysis request (retrieved by the POST transaction)

» Successful responses:

- Code: 202

  The analysis with ID 'resultid' is not complete and must be called again.

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

- Code: 200

```
{
  "entitiesanalysis": [
    {
      "type": "...",
      "concept": "...",
      "concept_norm": "..."
    }
  ],
  "resultid": "..."
}
```

-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)

-type: Type of entity identified (see available entity types below)

-entity: Entity found (person, country, company, etc.)

-entity_norm: Normalized version of the entity

Available entities types:

- **0:** Unknown
- **1:** Name of person
- **2:** Car license plate
- **3:** Place
- **4:** Phone number
- **5:** Email address
- **6:** Company
- **7:** Organization
- **8:** URL
- **9:** IP address
- **10:** Date
- **11:** Hour
- **15:** Money
- **16:** Address
- **17:** Twitter hashtag
- **18:** Twitter user
- **19:** Other alphanumeric

- **Example:**

  » POST Request:

```
-s -X POST --url https://svc02.api.bitext.com/entities/ -H "Authorization: beare
r 12345123451234512345123451234512345ab" -H "Content-Type: application/json
" --data '{"language":"eng", "text":"I have a beautiful house in Madrid"}'
```

» POST Response:


```
{
  "success": true,
  "message": "Request accepted",
  "resultid": "12345123451234512345123451234512345ab"
}
```


» GET Response:

```
curl -s -X GET --url https://svc02.api.bitext.com/entities/123456789abcdefg12
3456789abcdefg/  -H "Authorization: bearer 12345123451234512345123451
2345ab" -H "Content-Type: application/json
```

» POST Response:


```
{
  "entitiesanalysis": [
    {
      "type": "3",
      "entity": "Madrid",
      "entity_norm": "Madrid"
    }
  ],
  "resultid": "91e950cfe0454043b2cb51cd8b7650a4"
}
```

2.4 Categorization

The Bitext categorization service categorizes the sentences of a text according to a specific set of categories or a “coding plan.” A coding plan is a set of categories and words (or terms) that relate to the information you want Bitext to extract from your data. When you call the categories endpoint you must provide a coding plan identifier in the 'codingplanid' parameter. The identifier of the coding plan is its ‘key' attribute in the control panel.

When a text is sent to the categories endpoint, it is divided into different sentences. One or more categories is assigned to each sentence. In addition to the category assigned to the sentence, the service retrieves a list of terms that appear in the sentence that caused the sentence to be assigned to that particular category. For example, if the coding plan has the category ’House’ then all sentences that contain the word “house” are assigned to that category.

The category extraction service would therefore assign the sentence “I have a beautiful house in Madrid” to the category ‘House.’ If a sentences contains no words that trigger categorization, it is assigned to the default category ’None’ and the list of terms remains empty.

- **POST Request:**

  » Request:

  - Endpoint: /categories/
  - URL: https://svc02.api.bitext.com/categories/
  - Method: POST
  - Headers:

```
{
  "Authorization": "bearer [token]",
  "Content-Type": "application/json"
}
```

  - Data parameters:

```
{
  "language": "...",
  "text": "...",
  "codingplanid": "..."
}
```

-token: Your API credentials (can be copied from the My Profile section of the control panel)

-language: Language of the text you want to analyze (see available languages below)

-text: Text you want to analyze

-codingplanid: Identifier of the coding plan you want to use to categorize (the ‘key' attribute of the coding plan found in the control panel)

Available languages:

- **"eng":** English
- **"cat":** Catalan
- **"deu":** German
- **"fra":** French
- **"ita":** Italian
- **"nld":** Dutch
- **"por":** Portuguese
- **"spa":** Spanish

» Successful responses:

- Code: 201

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

- **GET Request:**

  » Request:

  - Endpoint: /categories/
  - URL: https://svc02.api.bitext.com/categories/:resultid/
  - Method: GET
  - Headers:

```
{
  "Authorization": "bearer [token]",
  "Content-Type": "application/json"
}
```

-token: Your API credentials (can be copied from the My Profile section of the control panel)

-resultid: Identifier of the analysis request (retrieved by the POST transaction)

» Successful responses:

- Code: 202

  The analysis with ID 'resultid' is not complete and must be called again.

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

- Code: 200

```
{
  "categoriesanalysis": [
    {
      "category": "...",
      "terms": [
        "..."
      ],
      "sentence": "..."
    }
  ],
  "resultid": "..."
}
```

-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)

-category: Category assigned to the current sentence

-terms: Words that triggered the categorization of the sentence

-sentence: Sentence in which the category was found

Available categories types:

    - **1001:** nominal
    - **1002:** adjectival
    - **1003:** adverbial
    - **1004:** verbal
- **Example:**

  » POST Request:

  -s -X POST --url https://svc02.api.bitext.com/categories/ -H "Authorization: bearer 12345123451234512345123451234512345ab" -H "Content-Type: application/json" --data '{"language":"eng", "text":"I have a beautiful house in Madrid", "codingplanid":"hotels_ENG_0"}'

  » POST Response:

  {

  "success": true,

  "message": "Request accepted",

  "resultid": "12345123451234512345123451234512345ab"

}

  » GET Response:

  curl -s -X GET --url https://svc02.api.bitext.com/categories/123456789abcdefg123456789abcdefg/ -H "Authorization: bearer 12345123451234512345123451234512345ab" -H "Content-Type: application/json

  » POST Response:

  {

  "categoriesanalysis": [

    {

      "category": "room",

      "terms": [

        "room"

      ],

      "sentence": "I have a beautiful room in Madrid"

    }

  ],

  "resultid": "1c20ccf9c0f24598a5ed09d23dfa460a"

}

3. Code Samples

5.1 Perl

```
use Mojo::UserAgent;
use Mojo::JSON qw(encode_json);
# User token, required for API access
# Replace this example token with your actual token. Your authentication token can be found in
# the My Profile section of the dashboard. Click API Credentials, copy the token and put it here.
my $oauth_token = "Example.Put_your_token_here";

# 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. Sentiment endpoint. PERL sample code\n";
print "-------------------------------------------------\n\n";

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

# Building the POST request to sentiment analysis endpoint
my $endpoint = "https://svc02.api.bitext.com/sentiment/";
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 can be found in the My Profile section of the dashboard. Click API Credentials, copy the token and use it in your code\n";
}


if ($post_result)
{
	print "Waiting for analisis 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 "Analisys results:\n\n";
	print encode_json $analysis;
	print "\n";
}
```

5.2 Python

```
import requests, json;


# User token, required for API access
# Replace this example token with your actual token. Your authentication token can be found in
# the My Profile section of the dashboard. Click API Credentials, copy the token and put it here.
oauth_token = 'Example.Put_your_token_here';


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


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


# Building the POST request to sentiment analysis endpoint
endpoint = "https://svc02.api.bitext.com/sentiment/";
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 applicabl
e
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 can be found in the My Profile section of the dashboard. Click API Crede
ntials, copy the token and use it in your code\n" );
```

```
if ( post_result ):
        print ( "Waiting for analisis 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 ( "Analisys results:\n\n" );
        print ( analysis );
        print ( "\n" );
```

5.3 Ruby

```
require 'httpclient'
require 'json'

# User token, required for API access
# Replace this example token with your actual token. Your authentication token can be found in
# the My Profile section of the dashboard. Click API Credentials, copy the token and put it here.
oauth_token = 'Example.Put_your_token_here'

# API request data: Language and text to be analyzed
user_language = 'enddg';
user_text = 'I have a beautiful house in Madrid';

puts '';
puts 'BITEXT API. Sentiment endpoint. PERL sample code';
puts '-----------------------------------------------';
puts '';

clnt = HTTPClient.new
# Building the POST request to sentiment analysis endpoint
endpoint = "https://svc02.api.bitext.com/sentiment/"
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 can be found in the My Profile section of the dashboard. Click API Credentials, copy the token and use it in your code'
end


if ( post_result )

        puts 'Waiting for analisis 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 'Analisys results:'
        puts ''
        puts analysis
        puts ''
end
```

5.4 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
        // Replace this example token with your actual token. Your authentication token can be found in
        // the My Profile section of the dashboard. Click API Credentials, copy the token and put it here.
  //public static String oauth_token = "Example.Put_your_token_here";
        public static String oauth_token = "Example.Put_your_token_here";

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

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

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

                // Sending the POST request
    URL urlPOST = new URL(endpoint);
    HttpURLConnection connectionPOST = createAPIConnection("POST",urlPOST);
    DataOutputStream outStream = new DataOutputStream(connectionPOST.getOutputStream());
    outStream.write(textdata.getBytes("UTF8"));
    outStream.flush();
    outStream.close();
                // Processing the response code of the POST request
                switch (connectionPOST.getResponseCode())
```

```
                        {
        case 201:  // 201 is the code for succesful request processing
                break;
                                                // 401 is the error code corresponding to an invalid token
        case 401:  System.out.println ( "Your authentication token can be found in the My Profile section of the dashb
oard. Click API Credentials, copy the token and use it in your code" );
                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 jResponse1 = new JSONObject(sResponse1);
      String action_id = jResponse1.getString("resultid");                // Identifier to request the analysis results
                        String post_msg = jResponse1.getString("message");                              // Error mess
age, if applicable
                        boolean post_result = jResponse1.getBoolean("success");                         // Success of
the request


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


                        if (post_result)
                        {
                                System.out.println ( "Waiting for analisis 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 ( "Analisys results:\n" );
                                System.out.println(sResponse2); // Print it with 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.getInputStream()));
        String decodedString;

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