Sentiment140

Search this site

API

We provide APIs for classifying tweets. This allows you to integrate our sentiment analysis classifier into your site or product.

Contents

Registering your application

You may register your application here: http://help.sentiment140.com/api/registration.

Please supply an `appid` parameter in the URL of your API requests.
The `appid` value should be an email address we can contact. For example, if you are using the JSON Bulk Classification API, the HTTP endpoint that you use to send requests may look like this:

```
http://www.sentiment140.com
/api/bulkClassifyJson?appid=bob@apple.com
```

where `bob@apple.com` is the main contact. You may URL-encode the `appid` value.

Technically you can use the API without supplying the `appid` parameter. But, we may block the requests that don't have an `appid` specified if we suspect abuse.

Commercial License

We offer a commercial license of our API for $200 per month. With this license, you classify up to 1 million tweets per day. We support credit card and invoice billing. If interested, please contact us.

Receiving updates

Please sign up on our mailing list to receive announcements about any changes.

Developer Documentation

Bulk Classification Service (JSON) - Recommended

This is the recommended way to utilize our API. You can send thousands of tweets per request, and receive the responses in bulk.

Requests
Requests should be sent via HTTP POST to `http://www.sentiment140.com/api/bulkClassifyJson`. The body of the message should be a JSON object. Here are some example requests:

Example 1 - Basic example
In this example, you send a JSON array inside a JSON object's "data" field. Each item should have a field called "text".

```
{"data": [{"text": "I love Titanic."},
          {"text": "I hate Titanic."}]}
```

Example 2 - Passing auxiliary fields
You can also include auxiliary fields ("id" in this case), which will be copied to the response. This is useful if you need to match the request items to the response items. Any data fields besides "text", "query", and "polarity" are not examined by the classifier and are simply copied to the response.

```
{"data": [{"text": "I love Titanic.", "id": 1234},
          {"text": "I hate Titanic.", "id": 4567}]}
```

Example 3 - Specifying the query
You can include a "query" parameter, which specifies what the tweet is about. It's recommended that you provide the query term. The query parameter helps us prevent certain keywords from influencing sentiment. For example, if the query was "Horrible Bosses" (referring to the 2011 movie), supplying the

query parameter would help us know that "horrible" is actually part of the query and shouldn't contribute to the calculated sentiment.

```
{"data": [{"text": "I love Titanic.", "query":
"Titanic", "id": 1234},
          {"text": "I hate Titanic.", "query":
"Titanic", "id": 4567}]}
```

Example 4 - Specifying a language

We currently support English and Spanish. We assume tweets are English by default, but you can force the Spanish model by supplying a language parameter:

```
{"language": "es",
 "data": [{"text": "Odio mi vida.", "query": "vida",
"id": 1},
          {"text": "Me encanta mi trabajo.", "query":
"trabajo", "id": 2}]}
```

If you want us to auto-detect the language, set the value of language to "auto":

```
{"language": "auto",
 "data": [{"text": "Odio mi vida.", "query": "vida",
"id": 1},
          {"text": "Me encanta mi trabajo.", "query":
"trabajo", "id": 2}]}
```

Example 5 - Specifying a topic

The words that people use to express sentiment can vary greatly between topics. For example, consider the word "scary." It's positive if you find *Silence of the Lambs* is scary, but negative if your Toyota's brakes are scary. By default, we use a generic sentiment model that works okay across different domains. But, if you supply a topic, we use a domain-specific classifier on the backend that can provide better accuracy.

In the API, include a "topic" parameter that specifies the subject domain of the tweet. Currently the only valid value for the "topic" parameter is "movies", but we are adding more.

```
{"data": [{"text": "I love Titanic.", "query":
"Titanic", "topic": "movies"},
          {"text": "I hate Titanic.", "query":
"Titanic", "topic": "movies"}]}
```

Response Body:
The response will be the same as the request, except a new field "polarity" added to each object. For example, the response for Example 2 above will look like the following:

```
{"data": [{"text": "I love Titanic.", "id":1234,
"polarity": 4},
          {"text": "I hate Titanic.", "id":4567,
"polarity": 0}]}
```

The polarity values are:

- 0: negative
- 2: neutral
- 4: positive

Example command-line usage:
Here's an example command you can run on a Linux shell that shows an example request and response:

```
$ curl -d "{'data': [{'text': 'I love Titanic.'}]}"
http://www.sentiment140.com/api/bulkClassifyJson
```

The response will look something like this:

```
{"data":[{"text":"I love Titanic.","polarity":4}]}
```

Simple Classification Service (JSON)

This API allows you to classify tweets individually via HTTP GET requests. The request looks like this:

```
http://www.sentiment140.com/api/classify?text=new+moon+is+awesome&
query=new+moon&callback=myJsFunction
```

Please see the Bulk Classification Service if you want to classify many tweets per request.

Request Parameters:

- text: The text you want to classify. This should be URL encoded.
- query: The subject. This should be URL encoded. (optional)
- callback: The callback function (optional). Leave this out if you want a pure JSON object returned.

Response Data:

- text: original text submitted

- query: original query submitted
- polarity. The polarity values are:
  - 0: negative
  - 2: neutral
  - 4: positive

Bulk Classification Service (CSV)

This makes it easy to use Linux's curl program to send tweets to be classified in bulk. You can send a **very simple text file** of tweets.

Example Request

1. Create a file called obama.txt with the following content:

```
obama is awesome
obama sucks
obama is eating a potato
```

2. Run the following command (one line):
   curl --data-binary @obama.txt "http://www.sentiment140.com /api/bulkClassify?query=obama"

   In case you're not familiar with curl, the --data-binary flag tells curl to preserve the new line character in the sent data set, and the @obama.txt tells curls which filename to read the data from.

Output:

The response will be a CSV with two fields:

- polarity. The polarity values are:
  - 0: negative
  - 2: neutral
  - 4: positive
- the text

Notes:

- The query parameter tells us what we should be classifying the sentiment towards. It's optional.
- This API should work for up to 10,000 tweets per call. You can perform multiple calls.

Subpages (1): API Registration

Comments

No puedes hacer un comentario sobre este documento.