© API проверки текстов на уникальность сервиса text.ru

Перед использованием API необходимо.

- Зарегистрироваться на сервисе text.ru (https://text.ru).
- Активировать свой аккаунт (пройти по ссылке, указанной в пришедшем на элетронную почту письме).
- Пополнить баланс (https://text.ru/finance?act=1) и либо купить пакет символов (https://text.ru/pack), либо приобрести PRO-аккаунт (https://text.ru/pro), который предоставляет возобновляемые ежедневно 50000 символов. Если с вашего аккаунта ранее не было регистраций в сервисе, будет единовременно предложено 15000 символов бесплатно.
- Получить ключ авторизации API_KEY на странице API проверки (https://text.ru/api-check).

Если для взаимодействия с API планируется использовать программы, написанные на php, можно пропустить следующие пункты описания собственно API, и перейти к описанию уже имеющейся билиотеки (https://api.text.ru/doc/lib/ru).

Протокол

- **Транспорт**: обмен данными между API с клиентским приложением осуществляется по HTTP-протоколу методом POST.
- **Точка входа**: зависит от конкретного типа запроса.
- **Запрос**: представляет собой массив опций, передаваемый API методом POST.
- **Ответ**: кодируется в JSON (содержимое зависит от вида запроса и его опций).

Ошибки

В случае, если на стороне сервера API возникла ошибка обработки запроса, будет возвращен ответ, содержащий два поля:

Поле	Тип	Описание
error_code	integer	Код ошибки
error_desc	string	Описание ошибки

Ошибки условно можно разделить на три категории:

Тип	Необходимо…
wait	просто подождать какое-то время, прежде чем повторно выполнять вызвавший её запрос (сервер не доступен, текст еще не проверен и т.п.)
skip	прекратить выполнять вызвавший её запрос (текст пустой или слишком короткий/длинный, ошибка проверки и т.п.)
stop	прекратить выполнять любые запросы к серверу (не хватает символов в пакетах, неправильный ключ API и т.п.)

В возвращаемом коде ошибки значащими являются только последние две цифры, т.е. коды вида 1xx аналогичны xx, например: ошибка с кодом 110 означает то же, что и ошибка с кодом 10.

Коды ошибок, их типы и описания:

Код	Тип	Описание
10	skip	Отсутствует проверяемый текст
11	skip	Проверяемый текст пустой
12	skip	Проверяемый текст слишком короткий
13	skip	Проверяемый текст слишком большой. Разбейте текст на несколько частей
20	stop	Отсутствует пользовательский ключ
21	stop	Пользовательский ключ пустой
40	wait	Ошибка доступа на сервере. Попробуйте позднее
41	stop	Несуществующий пользовательский ключ
42	stop	Нехватка символов на балансе
43	wait	Ошибка при передаче параметров на сервере. Попробуйте позднее
44	wait	Ошибка сервера. Попробуйте позднее
45	wait	Ошибка сервера. Попробуйте позднее
50	skip	Шинглов не найдено. Возможно текст слишком короткий
60	skip	Отсутствует проверяемый uid текста
61	skip	Uid текста пустой
70	stop	Отсутствует пользовательский ключ
71	stop	Пользовательский ключ пустой
80	stop	Текущая пара ключ-uid отсутствует в базе
81	wait	Текст ещё не проверен
82	skip	Текст проверен с ошибками. Деньги будут возвращены
83	wait	Ошибка сервера. Попробуйте позднее

Запросы

Все запросы **всегда должны содержать** в себе опцию userkey - API_KEY, служащий для авторизации на сервисе.

Формат описания запросов:

- Имя поля массива, передаваемого API методом POST.
- Тип передаваемого значение или указание конкретного значения (заключается в кавычки), которое нужно передать для того, чтобы опция сработала.
- Является ли опция обязательной (+ / –).
- Описание назначения поля.

Проверка баланса пакетов

Точка входа: `https://api.text.ru/account`.

Опция	Значение/Тип		Описание
`method`	`"get_packages_info"`	+	Указывает, какой метод выполнить `API`

Внимание! В целях безопасности, чтобы ваш IP-адрес не заблокировался, не рекомендуется обращаться к данному методу чаще 2 раз в минуту.

Отправка текста на проверку

Точка входа: `https://api.text.ru/post`.

Опция	Значение/ Тип		Описание
`text`	`string`	+	Проверяемый текст
`exceptdomain`	`string`	–	Домены, которые вы хотите исключить из проверки (через запятую или пробел)
`excepturl`	`string`	–	Ссылки, которые вы хотите исключить из проверки (через запятую или пробел)
`visible`	`"vis_on"`	–	Включение доступности результатов проверки другим пользователям
`copying`	`"noadd"`	–	Отключение сохранения результата проверки в архиве (ссылок вида `https://text.ru/antiplagiat/{text_uid}`)
`callback`	`string`	–	Callback (#11): URL (ссылка), на которую будет отправлен POST-запрос с результатами проверки

Получение результата проверки

Точка входа: `https://api.text.ru/post`.

Опция	Значение/Тип		Описание
`uid`	`string`	+	Уникальный идентификатор текста
`jsonvisible`	`"detail"`	–	Требование получить более детальную информацию о результатах проверки

Ответы

Проверка баланса пакетов

Поле	Тип	Описание
`size`	`integer`	Суммарный остаток символов по всем пакетам

Отправка текста на проверку

Поле	Тип	Описание
`text_uid`	`string`	Уникальный идентификатор текста

Получение результата проверки

Поле	Тип	Описание
`text_unique`	`float`	Уникальность текста в процентах с точностью до 2 знаков после запятой
`result_json`	`string`	Дополнительная информация о результатах проверки на уникальность в формате `JSON`
`spell_check`	`string`	Дополнительная информация о результатах проверки на правописание в формате `JSON`
`seo_check`	`string`	Дополнительная информация о результатах проверки на SEO-анализ в формате `JSON`

Поля `spell_check` и `seo_check` возвращаются только в том случае, если запрос был выполнен с опцией `jsonvisible`. Если проверка уникальности текста завершена, а проверка орфографии и SEO-анализ еще нет - соответствующие поля будут содержать пустую строку.

Информация об уникальности

Представляет собой массив/объект со следующими полями:

Поле	Тип	Описание
`date_check`	`string`	Дата окончания проверки текста на сервере
`unique`	`float`	Уникальность текста в процентах, с точностью до 2 знаков после запятой
`urls`	`array`	Список ссылок, по которым найдены совпадения с текстом
`clear_text`	`string`	Очищенный от служебных символов и знаков препинания ваш текст, состоящий из слов, разделенных пробелом

Каждый элемент массива ссылок `urls` содержит поля:

Поле	Тип	Описание
`url`	`string`	URL ссылки
`plagiat`	`float`	Процент совпадения текста по ссылке
`words`	`string`	Номера позиций "совпадающих" слов в тексте `clear_text` (через пробел, нумерация - от 0)

Поля `clear_text` и `urls[][words]` возвращаются только в том случае, если запрос был выполнен с опцией `jsonvisible`.

Пример:

```JSON
{
    "date_check": "29.03.2017 14:46:49",
    "unique": 0,
    "clear_text": "Wikipedia was launched on January 15 2001 by Jimmy Wales and Larry Sanger 11 Sanger coined its na
    "mixed_words": "4 10 29",
    "urls": [
        {
            "url": "https://en.wikipedia.org/wiki/Wikipedia",
            "plagiat": 100,
            "words": "0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 3
        }
    ]
}
```

Информация о правописании

Представляет собой массив, каждый элемент которого имеет следующие поля:

Поле	Тип	Описание
`error_type`	`string`	Тип ошибки ('Орфография', 'Пунктуация' и т.д.)
`reason`	`string`	Детальное описание ошибки
`error_text`	`string`	Текст фрагмента, в котором обнаружилась ошибка
`replacements`	`array`	Список предлагаемых вариантов замены (может быть пустым)
`start`	`integer`	Начальная позиция фрагмента, в котором найдена ошибка
`end`	`integer`	Конечная позиция фрагмента, в котором найдена ошибка

Пример:

```JSON
[
    {
        "error_type": "Spelling",
        "replacements": [
            "milk",
            "mild",
        ],
        "reason": "Spelling error found",
        "error_text": "mildd",
        "start": 255,
        "end": 272
    },
    {
        "error_type": "Capital letters",
        "replacements": [
            "Hello"
        ],
        "reason": "This sentence doesn't start with a capital letter",
        "error_text": "hello",
        "start": 276,
        "end": 287
    }
]
```

Информация по SEO-анализу

Представляет собой массив/объект со следующими полями:

Поле	Тип	Описание
`count_chars_with_space`	`integer`	Количество символов с пробелами
`count_chars_without_space`	`integer`	Количество символов без пробелов
`count_words`	`integer`	Количество слов в тексте
`water_percent`	`float`	Процент "воды"
`spam_percent`	`float`	Процент заспамленности
`mixed_words`	`array`	Номера позиций слов в тексте, у которых присутствуют символы из разных алфавитов и имеющих схожее написание. Нумерация начинается с 0
`list_keys`	`array`	Список ключей в тексте, отсортированных по частоте встречаемости
`list_keys_group`	`array`	Список ключей в тексте, отсортированных по числу значимых слов и сгруппированных по составу слов

Каждый элемент списка `list_keys` содержит поля:

Поле	Тип	Описание
`key_title`	`string`	Текст ключа
`count`	`integer`	Количество вхождений ключа в текст во всех формах

Каждый элемент списка `list_keys_group` содержит поля:

Поле	Тип	Описание
`key_title`	`string`	Текст ключа
`count`	`integer`	Количество вхождений ключа в текст во всех формах
`sub_keys`	`array`	Список подключей - структур, аналогичных элементам списка `list_keys` .

Пример:

```JSON
{
    "count_chars_with_space": 620,
    "count_chars_without_space": 545,
    "count_words": 77,
    "water_percent": 11,
    "spam_percent": 38,
    "mixed_words": [8, 11, 47],
    "list_keys": [
        {
            "key_title": "check",
            "count": 6
        },
        {
            "key_title": "text",
            "count": 5
        },
        {
            "key_title": "antiplagiat",
            "count": 3
        }
    ],
    "list_keys_group": [
        {
            "key_title": "check text",
            "count": 3,
            "sub_keys": [
                {
                    "key_title": "check",
                    "count": 6
                },
                {
                    "key_title": "text",
                    "count": 5
                }
            ]
        }
    ]
}
```

Callback

Проверка текста на уникальность занимает определенное (заранее неизвестное) время. Если вы хотите получить уведомление о факте завершения (вместе с результатами) проверки вашего текста, то вы можете при добавлении текста на `API` в параметре `callback` указать адрес вашего скрипта, который обработает наш `POST`-запрос с результатами проверки текста.

Поля запроса аналогичны тем, которые получаются запросом получения результата проверки, но имеются следующие отличия:

- запрос будет содержать поле `uid` - уникальный идентификатор текста;
- вместо поля `result_json` будет поле `json_result`, аналогичное по структуре и смыслу (без полей `clear_text` и `urls[][words]`, как при отсутствии опции `jsonvisible`);
- будет присутствовать поле `spell_check`, но отсутствовать `seo_check`.

Максимальное число попыток отправки результатов проверки - **3**. Если проверка на правописание закончится позже проверки на уникальность, то будет отправлено 2 запроса по мере окончания проверок (во время первого запроса поле `spell_check` будет равно пустой строке).

Чтобы убедиться, что вы успешно получили и обработали результат, на вашей странице должно отобразиться **ok** и ничего более.

Примеры

PHP

В PHP запрос можно отправить одним из следующих способов (если не использовать предоставляемую нами билиотеку (https://api.text.ru/doc/lib/ru)): при помощи функции file_get_contents (https://www.php.net/manual/ru/function.file-get-contents.php)

```php
$request = http_build_query([
    'userkey' => 'my-api-key',
    'method'  => 'get_packages_info',
]);

$opts = ['http' => [
    'method'  => 'POST',
    'header'  => 'Content-Type: application/x-www-form-urlencoded',
    'content' => $request
]];

$context  = stream_context_create($opts);

$response = file_get_contents('https://api.text.ru/account', false, $context);
```

или расширения curl (https://www.php.net/manual/ru/book.curl.php):

```php
$request = http_build_query([
    'userkey' => 'my-api-key',
    'method'  => 'get_packages_info',
]);

$options = [
    CURLOPT_URL             => 'https://api.text.ru/account',
    CURLOPT_RETURNTRANSFER  => true,
    CURLOPT_POST            => true,
    CURLOPT_POSTFIELDS      => $request,
];

$ch = curl_init();
curl_setopt_array($ch, $options);
$response = (string) curl_exec($ch);
curl_close($ch);
```

или библиотеки guzzlehttp/guzzle (https://packagist.org/packages/guzzlehttp/guzzle):

```php
$request = [
    'userkey' => 'my-api-key',
    'method'  => 'get_packages_info',
];

$client = new GuzzleHttp\Client();

$response = $client->request(
    'POST',
    'https://api.text.ru/account',
    [GuzzleHttp\RequestOptions::FORM_PARAMS => $request]
);
```

В всех случаях результатом должна быть `json`-строка:

```php
$result = @json_decode($response, true);
if (JSON_ERROR_NONE !== json_last_error()) {
    // Ошибка разбора JSON.
} elseif (empty($result) || !is_array($result)) {
    // Неправильный формат ответа.
} else {
    var_export($result);
    // Будет выведено:
    // array (
    //   'size' => 24698146,
    // )
}
```

JS

Для отправки запросов `JavaScript`-ом можно воспользоваться, например, jQuery.ajax() (https://api.jquery.com/jquery.ajax/):

```javascript
var request = {
    userkey: 'my-api-key',
    method:  'get_packages_info'
};

$.ajax({
    url:      'https://api.text.ru/account',
    type:     'post',
    dataType: 'json',
    data:     request
    error:    (jqXHR, textStatus, errorThrown) => {
        console.error(jqXHR, textStatus, errorThrown);
    }
    success:  (response, textStatus, jqXHR) => {
        console.info(response);
        // Будет выведено:
        // {"size":24698146}
    }
});
```

Python

```python
import requests

try:
    request = {
        'userkey': 'my-api-key',
        'method':  'get_packages_info'
    }
    response = requests.post('https://api.text.ru/account', data = request).json()
    print(response)
    # Будет выведено:
    # {u'size': 24698146}
except requests.exceptions.RequestException as ex:
    print('ERROR: %s' % ex)
```

Ruby

```ruby
require 'net/http'
require 'uri'
require 'json'

begin
    uri      = URI.parse("https://api.text.ru/account")
    request  = {
        "userkey" => "my-api-key",
        "method"  => "get_packages_info"
    }
    response = Net::HTTP.post_form(uri, request)
    result   = JSON.parse(response.body)
    puts result
    # Будет выведено:
    # {"size"=>24698146}
rescue => ex
    puts ex
end
```

1C

Пример использования `API` можно посмореть на сайте INFOSTART.RU (https://infostart.ru/) в публикации Автоматизация проверки на плагиат в ИС "Курсовые работы" (https://infostart.ru/public/878398/) (опубликовано сторонним разработчиком - не сотрудником text.ru (https://text.ru/)).

Рекомендации

Проверка документов

Для проверки файлов можно использовать пакет text-media/file-parser (https://packagist.org/packages/text-media/file-parser), поддерживающий возможность поиска текста в файлах следующих типов: `doc`, `docx`, `epub`, `fb2`, `html`, `odt`, `pdf`, `rtf`.

Организация очередей

Для организации очередей проверки текстов можно использовать RabbitMQ (https://www.rabbitmq.com/), примеры использования которого в `PHP` доступны на офсайте (https://www.rabbitmq.com/tutorials/tutorial-one-php.html). Такой подход будет оптимальней, нежели периодическое сканирование базы на наличие требующих проверки текстов. Для работы с такими очередями моджно использовать библиотеку php-amqplib/php-amqplib (https://github.com/php-amqplib/php-amqplib) (или php-amqplib/rabbitmq-bundle (https://github.com/php-amqplib/RabbitMqBundle) для Symfony (https://symfony.com/)).

Создание демонов

Простейшим способом автоматической проверки нуждающихся в этом текстов является настройка планировщика задач, однако оптимальней будет "слушать" очередь (см. предыдущий раздел) постоянно работающим скриптом-"демоном". Это можно сделать при помощи Supervisor (http://supervisord.org/), который может "демонизировать" скрипт на любом языке, а также включает в себя `API` для удаленного управления "демонами" и `web`-интерфейс. Для "демонизации" своего скрипта достаточно в папку `/etc/supervisor/conf.d` добавить свой `conf`-файл вида:

```ini
[program:plagiarism-api]
user=my-user-name
command=/my-daemon-folder/my-plagiarism-api-daemon.php
autostart=true
process_name=%(program_name)s-%(process_num)02d
numprocs=1
redirect_stderr=true
stdout_logfile=/my-daemon-folder/log/%(program_name)s-%(process_num)02d.log
stdout_logfile_maxbytes=10MB
stdout_logfile_backups=3
```

Сам же скрипт "демона" должен иметь в простейшем случае примерно следующий вид:

```php
declare(ticks=1);

$isRunned = true;

pcntl_signal(SIGTERM, function() {
    global $isRunned;
    $isRunned = false;
});

while ($isRunned) {
    // "Слушаем" очередь и отправляем текст из нее на проверку.
}
```

Использование `callback`

Можно, как и в случае отправки текста на проверку, использовать планировщик задач или "демон", но это бессмысленная и беспощадная трата своих ресурсов, да и текст от постоянного обращения к `API` быстрее не проверится.

Оптимальным вариантом будет создать скрипт, который срабатывает в момент, когда сервер `API` завершил проверку, сформировал и отправил уведомление об этом (включая результаты проверки) владельцу по адресу, который он указывал в параметре `callback` при отправке текста на проверку.

Мониторинг баланса

См. Zabbix (https://www.zabbix.com/ru/) - можно настроить так, чтобы он получал остаток баланса по пакетам символов и уведомлял владельца в том случае, если баланс низок.

Для этого он должен уметь отправлять запросы на `API` - это можно сделать, если использовать имеющуюся билиотеку (https://api.text.ru/doc/lib/ru) - она содержит необходимую консольную утилиту.