Professional Bachelor Applied Information Technology

Responding to support tickets in an unfamiliar language

Jonathan Lauwers

Promoters:

Mitch Dries
Johan Cleuren

Otys Prague
PXL University of Applied Sciences and Arts Hasselt

Bachelor paper Academic year 2018-2019

Professional Bachelor Applied Information Technology

Responding to support tickets in an unfamiliar language

Jonathan Lauwers

Promoters:

Mitch Dries — Otys Prague

Johan Cleuren — PXL University of Applied Sciences and Arts Hasselt

Bachelor paper Academic year 2018-2019

Acknowledgements

After an unforgettable three years of studying at PXL University College, I am very proud to present my thesis. During these three years I have learned and experienced more than I could have ever imagined. From programming in Assembly to doing my internship in Prague. All of these experiences could not have been possible without the help from certain people of which I would like to thank a few.

First and foremost, I would like to thank Otys for a very enjoyable internship at their office in Prague. I learned a lot of new topics and met some very nice people along the way. Specifically, I would like to thank Mr Mitch Dries, for being my internship promoter and helping me whenever I had a question. I would also like to thank Mr Jirka Bruijn for making this internship at Otys possible.

Next, I would like to thank my school promoter, Mr Johan Cleuren, for guiding and helping me when needed and providing valuable feedback on my thesis topic and research.

I would also like to thank Mr Johan Cleuren one more time, as well as Ms Marijke Sporen, for allowing me to study and do my internship abroad. Both experiences resulted in memories I will never forget. It would not have been possible without your help.

Lastly, I would like to thank my family, friends and girlfriend for the continuous support and guidance throughout this journey.

Abstract

Otys, being a major recruiting software company, encounters a wide variety of nationalities. This comes with certain obstacles, such as different customers speaking different languages which not everyone understands. Thinking about how to overcome this obstacle, Otys thought of a solution featuring Google's Cloud Translation API.

Otys has the option to send support tickets, which can be done by any customer at any time. This is where the Google Translate functionality comes into play. Once a support ticket has been sent, that support ticket now has the option to be translated automatically. This functionality can be used by clicking the Google Translate button at the top right corner of the support ticket. Clicking this button detects the language of the ticket, retrieves your current profile language and translates the ticket to your current profile language.

This whole process is realized with the use of Elasticsearch, a RESTful search and analytics engine to store and retrieve data in an efficient and expandable way, and Google's Cloud Translation API, a language detection and translation API which supports over 100 different languages. When the button is pressed, a call to the Elasticsearch storage is made, checking if the translation has already been made. If so, that translation is taken and presented to the user. If not, the translate API is called to translate the text, the translation is stored in the Elastic Search storage and is then presented to the user.

If users reply to the ticket after having the Google Translate feature enabled, they are able to do so in their own preferred language as the reply is automatically translated to the original ticket's language once it is sent.

Currently, the language detection and translation section of this thesis are performed by using Google's Cloud Translation API. However, is this the best option for Otys? To answer this question, the research topic focuses on the alternatives of Google's Cloud Translation API. The research will cover the speed, accuracy and price of every alternative with a critical comparison in the end. Using this gathered information will reveal which alternative suits Otys best.

Table of contents

List of figures

List of tables

List of abbreviations

API: Application programming interface

Index: Comparable to a database in a relational database. It has mapping which defines multiple types.

CMS: Content Management System

RbMT: Rule-based machine translation

SMT: Statistical machine translation

NMT: Neural machine translation

MT: Machine translation

AI: artificial intelligence

Introduction

Nowadays, people are more connected than ever. With everyone being connected to each other, it is no surprise that people get into contact with people from all over the world. Social media is a perfect example of this, as people can use it to connect and communicate with people from all over the world with a simple publish of a social media post.

This is not only the case for people, but also for companies. With the possibility of communicating worldwide with software like Slack and Skype, companies can now have customers from all over the world and communicate with them as if they were in the same room. Keeping this in mind, this obviously brings some challenges with it.

One of the challenges communicating with people from all over the world is language. This challenge is no surprise, as not everyone speaks the same language. Even with English being so widely promoted, it is not understood or used by everyone in a multinational company environment. Therefore, companies like Google created Google Translate to remove this language barrier between different speaking people.

One of the companies that faces this language barrier problem is Otys. This is because Otys has a wide variety of clients who all have the possibility to send support tickets. These support tickets are often written in the customer's native language, meaning that not all developers can understand what is written in the support ticket and must use external services like Google Translate to figure out what the support ticket is about.

In this thesis, the possibilities of solving and/or simplifying this whole process are investigated. How it can be done and what is required to solve this language barrier in the most elegant way, are the main problems that are discussed. Besides this, there is research provided regarding the competitors of Google's Cloud Translation API and if Google's API is actually the best choice for Otys.

I. Traineeship report

1 About the company

Otys is an international company, specialised in the creation of recruitment software. They have over 15 years of experience with over 1200 clients and offices located in The Netherlands, France, Belgium and Czech Republic, resulting in a service that is available worldwide.

Otys has one main application, “Go!”, which is created to simplify the client’s recruitment work. The application gives the client the option to post their own job applications to their own website or multipost in just a few clicks and has a semantic search and match feature to easily link multiple candidates to a job application based on the client’s criteria. The application is web based, meaning that it requires no download and is mobile and tablet friendly.

Besides Otys’s main application, they also create custom websites for clients and have created over 1000 job application and company websites.

2 Introduction internship subject

2.1 Responding to support tickets in an unfamiliar language

Otys, being an international company, faces a wide variety of nationalities on a daily basis. This comes with the obstacle of people being unable to communicate with each other because they speak different languages. To solve this problem, Otys thought of implementing Google's Cloud Translation API.

The first situation in which the translate feature is implemented is the support feature. The reason why Otys chose the support feature is because this is where many clients send their questions and/or bugs, which are often written in the native language of the client. This results in the developers from Otys not always being able to understand what the ticket is about. This makes the support feature a perfect place to implement the translate feature which translates the support tickets to the user's current profile language.

2.2 What must be implemented

2.2.1 Settings

To create a very customisable and user-friendly experience, there are four different settings related to the translate feature.

The first setting is a setting to enable the translate feature in general. When this feature is disabled, the translate button will not work and return a message telling the user the translate feature is disabled. When this feature is enabled, the user has the possibility to translate the currently selected support ticket to their own profile language by clicking the Google Translate button.

Secondly, there is a setting to enable the automatic translating of the support ticket. This means that when a support ticket is loaded while the automatic translating setting is enabled, the ticket is automatically translated to the profile language of the user. To use the automatic translating feature, the general translate setting must also be enabled.

Thirdly, there is a setting to enable the saving of answers in the original language of the ticket. With this feature enabled, the user can reply to any ticket in their preferred language. Their answer is then automatically translated to the original language of the ticket. This means that if a ticket is written in French and the user replies in Dutch, the answer is automatically translated and saved to French. To use this feature, the general translate setting must also be enabled.

Lastly, there is a setting to exclude certain languages from being translated to provide the option of not automatically translating certain languages. Before every translation, the exclude languages setting is checked and compared to the language the ticket must be translated to. If for example, the user selects English to be excluded, no English ticket is translated to the profile language of the user. To use this feature, the general and automatic translate setting must be enabled.

2.2.2 Elasticsearch

Elasticsearch is a RESTful search and analytics engine capable of storing high amounts of data. The reason Elasticsearch is used, is to store translations and avoid unnecessary translations with Google's Cloud Translation API. This is to reduce the cost of using Google's API, as they work with an amount-of-character-based payment system. So instead of using Google's API every time a support ticket is translated, the translation is searched in the Elasticsearch index. If the translation is not found, the message is translated using Google's API and is then stored in the Elasticsearch index in case the translation is requested in the future.

2.2.3 Database table

To create a more customisable automatic translate feature, there is a database to hold the 'mode' of certain tickets. This mode is only relevant when the automatic translate setting is enabled as this mode allows the automatic translate feature to be disabled for the currently selected support ticket.

When the translate button is pressed for the first time, with the automatic translate setting enabled, a row in the 'gt_translate_mode' table is created with a variety of data relevant to the ticket, but most importantly, with the mode set to "OFF". This means that the automatic translate feature of the currently selected support ticket is disabled and returns the original support ticket instead of the automatic translated ticket. Pressing this button again switches the mode to ON, which re-enable automatic translating of that support ticket and displays the translated ticket.

2.2.4 Google's Cloud Translation API

To translate the support tickets, a service for language detecting and text translating is needed. When it comes to text translations, Google's Cloud Translation API is the most known API service due to its popular and worldwide use. Therefore, Otys proposed to use this API. It provides both needed services, language detection and text translation, in one.

3 Start internship subject

3.1 The database

Otys has a bit less than thirty different databases with a lot of valuable information. To insert, manipulate or delete an entry in one of the tables, Navicat is used. The reason Navicat is used, is because it provides a clear view of all the data inside the selected database.

However, to ensure no major mistakes are made when manipulating a database or table, Otys uses a migrating service called Otys Ruksak. This is a migrating service made for Otys to ensure every SQL query made to the databases is done safely and correctly.

A benefit of Otys Ruksak is that it enables version control, as the queries performed are only applied to certain databases while other databases are left untouched, enabling the untouched databases to be used in case a backup is needed.

Figure 1: Otys Ruksak page

When a database has be to created, altered or removed, it is possible to do so by creating a new migration. This migration holds four different functions, 'getLoadEstimateUp', 'getLoadEstimateDown', 'up' and 'down'.

The first function is the 'getLoadEstimateUp function'. This function requires a value from one to ten, and bases the server intensity on the value provided. If a low value is given, it informs the server a simple query is provided. The server then executes the query at a lower intensity. If a high value is given, this informs the server a very complex and heavy query is provided. The server then schedules the query later in the day. This is to avoid interference with other databases as a high intensity query might result in the temporary melting of the servers which could make some data temporarily unavailable. However, this function is usually not used.

The second function is the 'up function'. This is where developers can write queries by providing a database and a query. The provided query has to be an SQL query in order to be executed. When the migration is executed and the query contains a wrong syntax, Otys Ruksak detects this and prevents the query from being executed.

Next, is the ‘getLoadEstimateDown and down function’. These functions are the exact opposite of the previous two, and are called when the migration has to be reverted. However, developers usually do not provide much attention to these functions as the migration is supposed to be permanent and should not be undone. However, even without a proper ‘getLoadEstimateDown and down function’, the reverting of the migration can still be done easily because of Otys Ruksak’s version control.

To create the table that was required for the support ticket’s translate feature, a simple create table function is executed with the data fields required.

```
class GoogleTranslationTable extends Ruckusing_OtysMigration implements Ruckusing_iHeavyweight {

        public function getLoadEstimateUp() { // please estimate how server-intensive the up() operations will be (1 is "no load at all", 10 is "it will melt the server")
                return 1;
        }

        public function up() { // add the actions you want to apply
                $this->assert_database('otys_tables');
$this->execute("CREATE TABLE `gt_translation_code` (`id` int(11) NOT NULL,
  `id_client` int(11) NULL,
  `id_user` int(11) NULL,
  `entity` int(11) NULL,
  `id_entity` int(11) NULL,
  `mode` enum('ON','OFF') NULL DEFAULT 'OFF',
  `dt_set` datetime(6) NULL ON UPDATE CURRENT_TIMESTAMP(6),
  PRIMARY KEY (`id`),
  CONSTRAINT `id_entity` FOREIGN KEY (`id_entity`) REFERENCES `otys_tables`.`Entities` (`id`)
);");


        }//up()

        public function getLoadEstimateDown() { // please estimate how server-intensive the down() operations will be
                return 1;
        }

        public function down() { // add the opposite of your actions here
                $this->assert_database('otys_tables');
        }//down()

}
```

Figure 2: Query to create the support ticket’s database

3.2 Elasticsearch index

3.2.1 Installing Elasticsearch

To get started, Elasticsearch is installed in the project. To do this, a require is added to the ‘composer.json’ file.

```
{
    "require": {
        "elasticsearch/elasticsearch": "~2.0"
    }
}
```

Figure 3: Require needed for Elasticsearch

Next, the client is installed with composer.

```
curl -s http://getcomposer.org/installer | php
php composer.phar install --no-dev
```

Figure 4: Elasticsearch composer install

3.2.2 Creating an index

With Elasticsearch installed, an index can be created very easily.

```
private function getClient()
{
    if (is_null($this->client)) {
        $this->client = ClientBuilder::create()
            ->setHosts(CoreConfig::$elasticSearchConnections)
            ->setSelector( selector: static::CONNECTION_SELECTOR_CLASS)
            ->build();
    }

    return $this->client;
}
```

Figure 5: Elasticsearch client instantiate function

First, a client is created with the 'ClientBuilder's create function'. This function creates and returns a new instance of the 'ClientBuilder class'.

Next, the host is set. This can be done by providing one or more IP addresses to the Elasticsearch cluster, which is then used by the client. These IP addresses are purchased from Elasticsearch and are in this case provided by Otys.

With the hosts provided, the 'setSelector function' is called, which chooses one of the provided hosts randomly. This is provided by Otys as Otys has multiple IP addresses available and uses the random selector to distribute the data evenly.

Lastly, with all this data set, the build function is called which builds the Elasticsearch client.

Now that the client is instantiated, the index can be created. This is done by creating an array with four values:

- index
- type
- id (optional)
- body

```
public function create($body){
    $req = [];
    $req['index'] = static::TR_INDEX;
    $req['type'] = static::TR_TYPE;
    $req['body'] = $body;

    return $this->getClient()->index($req);
}
```

Figure 6: Elasticsearch 'create function'

The value provided to ‘index’ is the name of the created index. In this case, the static name ‘google_translate’.

The ‘type’ value receives the static value of ‘translations’, as translations are what is stored into the created index.

‘id’ is an optional value. If no ID is provided, an automatically generated ID is created and given to the created index. If a custom ID is provided, this ID is used. This ID is later used to retrieve data.

Lastly, the ‘body’ value, which stands for ‘document body’, is an associative array with key value pairs corresponding to the data in the document. This is where an array of translation data is inserted, which is later used to search for already made translations.

```
public function elasticSearchStore($origLang, $targetLang, $entity, $entityId, $orig_hash, $translation){

    $gtc = new GoogleTranslateClient();

    $data = array(
        'id_client' => $this->clientid,
        'language_orig' => $origLang,
        'language_to' => $targetLang,
        'entity' => $entity,
        'id_entity' => $entityId,
        'orig_hash' => $orig_hash,
        'translation' => $translation,
    );

    $reply = $gtc->create($data);

    return $reply;
}
```

Figure 7: Body of the Elasticsearch index

With the creation of this index, storing and searching of data is now available.

3.2.3 Searching the index

To search the created index for entries, a ‘search function’ is created. This function uses the same index and type as the create function, being ‘google_translate’ and ‘translations’. Besides the index and type, a search parameter needs to be provided. In this case, an MD5 hash is used to look for matching hashes. The parameter name for the index’s MD5 hash is ‘orig_hash’, which is what is needed when searching. The required query to search the ‘orig_hash’ can be seen in the image bellow.

```
public function search($body)
{
    $req = [];
    $req['index'] = static::TR_INDEX;
    $req['type'] = static::TR_TYPE;
    $req['body']['query']['match']['orig_hash'] = $body;

    return $this->getClient()->search($req);
}
```

Figure 8: Elasticsearch ‘search function’

3.3 Settings

3.3.1 Creating the setting

The creating of settings is done by adding rows to the 'settings_props' table of the 'Otys_tables' database. This database creates basic settings based on a static template and has the option to contain a simple enable/disable button or a simple dropdown list. To create a setting, a variety of information must be provided.

- id: A unique value that can later be used to assign the made setting to a desired category.
- group_id: A value from 1 to 62 that assigns the setting to the chosen predefined group.
- code: A code that can be used to programmatically retrieve the value of that setting.
- name: Name that is shown in the settings menu.
- prop_type: Defines what prop type the setting has. This can be an 'int, bool or string'.
- defval: The default value of the setting.
- use_type: To define if the setting is for clients, users or both.

id	group_id	code	name	prop_type	defval	portable	expiration	use_type
3056	58	google_translate_save_origir	Go! Google Translate - save	bool	0	1	0	user
3058	58	google_translate_auto_trans	Go! Google Translate - auto	bool	0	1	0	user
3060	58	google_translate_exclude_la	Go! Google Translate - auto	string	0	1	0	user

Figure 9: 'settings_props' examples of translate settings

The only exception out of the four settings, is the exclude language settings. This is the only setting that requires a list instead of an enable/disable button. This is done by changing the 'prop_type' to string and adding extra information to the 'logical_subtype and 'cdt_class' of that setting. This extra information is only necessary for this kind of setting.

- Logical_subtype: The kind of subtype. Options are 'list, date and list_multiple'.
- cdt_class: The class the data must be fetched from.

To create this setting, the 'logical_subtype' is changed to 'list_multiple'. This creates a dropdown list of the received data from the 'cdt_class'. To get the right data from the 'cdt_class', a link to the 'CmsLanguages' class is made. This class returns a list of all the CMS languages that need to be selected from.

```
class CmsLanguages extends AbstractCdtMultiSelect implements ICustomDataType {

    public function getListValues() {
        $db = $this->getCurrentClientDb();
        return $this->getIndexedValues(
            $db,  q: "select
                    id,
                    language_en
                from otys_cms.Languages
                order by id"
        );
    }
```

Figure 10: CmsLanguages class

3.3.2 Linking the setting

To give the settings its own category, a category is added in the 'settings_categories' table. In this table, a category is added by simply adding an entry with an original ID, the ID of the parent if the category must become a subcategory, and a name.

To create a separate category named 'Google Translate', an entry is added with a unique ID and no parent ID. This is to ensure the Google Translate category has no parent and becomes its own category.

To make this category a bit more clear, a subcategory is added with the name 'Basic settings'.

id	id_parent	name
855	0	Google Translate
856	855	Basic settings

Figure 11: settings_categories table Google Translate and Basic Settings row

With these categories added, the new settings are linked to the new category by creating new records with an original ID, the ID of the setting and the ID of the category.

id	id_property	id_category
6918	3053	856
6919	3056	856
6920	3058	856
6921	3059	856

Figure 12: settings_props_categories linking of settings

By linking the settings to the category, the linked settings are automatically added to the settings menu.

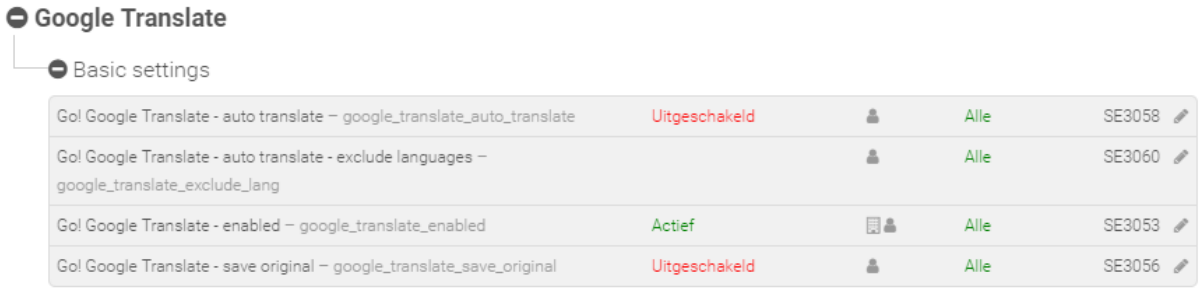

Figure 13: Google Translate settings

3.4 Google's Cloud Translation API

3.4.1 Integration

To integrate Google's Cloud Translation API into the Otys app, a Cloud Platform project is needed. On this platform, Otys has a project named 'Otys', which is required for Google's Cloud Translation API to be integrated.

With a project set up, a composer require command is executed to get the required files to connect with Google's API.

```
composer require google/cloud-translate
```

Figure 14: Composer install required to use Google's Cloud Translation API in PHP

3.4.2 Usage

With these files obtained, it is possible to use the Google Translate API.

First, a function to instantiate a Google Translate Client is created. In this function, a new object of the class 'TranslateClient' is instantiated with the 'key' value as a parameter. This key is generated by enabling the API in the console of the created Google Cloud Platform project and enables translating text and detecting languages.

```php
protected function instantiateGoogleTranslateClient()
{
    $translateClient = new TranslateClient([
        'key' => $this->key
    ]);

    return $translateClient;
}
```

Figure 15: Function to instantiate Google's Cloud Translation API

With this function, it is possible to detect languages and translate text by using the instantiate function, followed by the desired function.

3.5 The translate button

3.5.1 layout changes

To translate a support ticket, a button is needed. This button is a small button containing the Google Translate logo, created by Amanda Diana, and is located at the top right corner of the selected ticket. Adding this to the front end required one line of HTML code and five lines of CSS code.

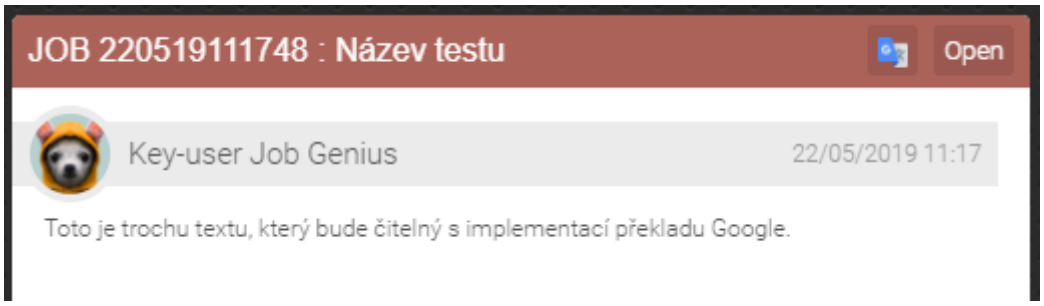

Figure 16: Support ticket with Google Translate button

The translate button uses the 'isTranslated' boolean defined in the controller, to determine the layout of the button. If the 'isTranslated' boolean is false, the button shows a "normal" layout as shown in Figure 16. If the 'isTranslated' boolean is true, the button shows a "pressed" layout as shown in Figure 17.

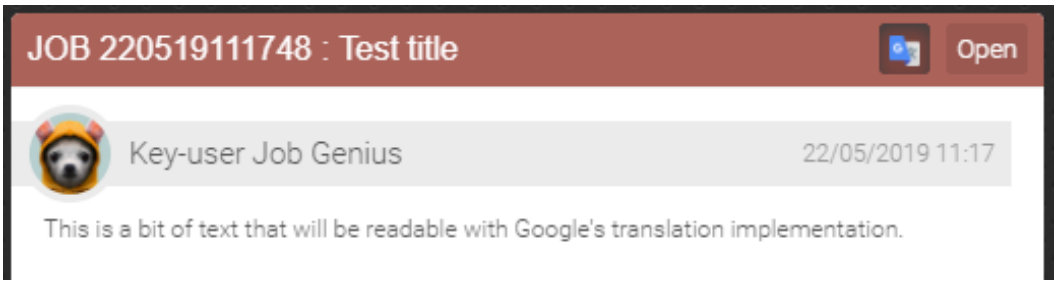

Figure 17: Translated support ticket with Google Translate button

When opening the selected ticket, another Google Translate button can be found at the top right corner, which also uses the 'isTranslated' boolean to determine its layout as shown in Figure 18. Both buttons have the functionality of translating the currently selected ticket when clicked.

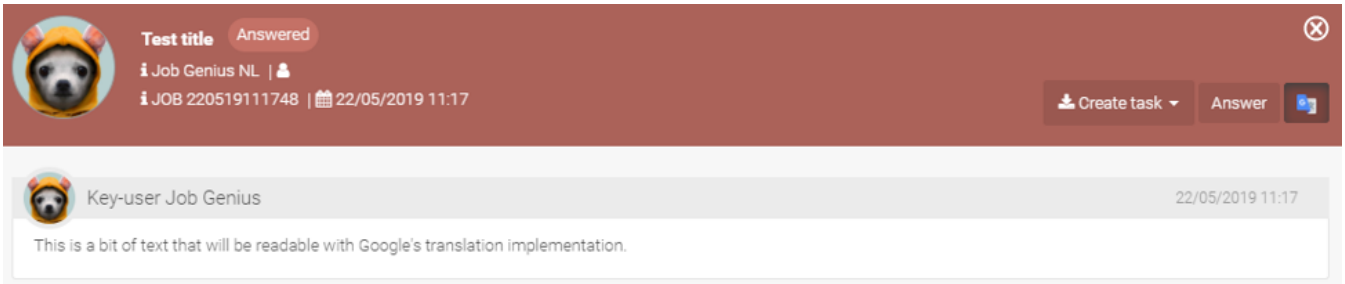

Figure 18: Detail view of a translated support ticket

3.5.2 Support controller

This button is linked to the 'translate function', which can be found in the controller of the support feature. This controller has the responsibility of loading all tickets for the current user. In the controller, a function is created with the name 'translate' which receives a ticket. When the function is called, the 'ticket, entity and entityId' are forwarded to the service. This service returns the translated ticket which is then swapped with the original ticket.

```
$scope.translate = function(ticket) {
    ows('GoogleTranslateService.translateTicket', $scope.entity, $scope.entityId, ticket)
        .then(function (res) {
            ticket.subject = res.ticket.subject;
            ticket.messages = res.ticket.messages;
            $scope.isTranslated = res.isTranslated;
        })
}
```

Figure 19: 'Translate function'

Besides the 'translate function', a 'translateOnPageLoad' function is added to the 'load function' of the support view. This function sends the currently loaded 'ticket, entity and entityId' to the service when the page is loaded. If the settings are enabled, the function returns a ticket with a translated subject and messages.

```
.then(function (res) {
    ows('GoogleTranslateService.translateOnPageLoad', $scope.entity, $scope.entityId, res)
        .then(function(res){
            $scope.ticket = res.ticket;
            $scope.isTranslated = res.isTranslated;
    });

});
```

Figure 20: 'TranslateOnPageLoad function'

Detail view

In the detailed view of the selected support ticket, the 'translate and translateOnPageLoad functions' are both used in the same way as the support view. However, in the detailed view, the user can answer to the selected support ticket.

To enable correct answering, a separate function is created to verify if the save original setting is enabled. If this setting is enabled, the reply is automatically translated to the original language of that ticket and then added to the ticket. If not, the function returns the sent answer and adds that to the ticket.

```
return ows('GoogleTranslateService.saveOriginalTicket', $scope.entity, $scope.entityId, $scope.ticket, ticketData).
    then(function (res) {
        return ows('SupportService.addResponse', $scope.ticket.uid, res)
```

Figure 21: Detailed view 'SaveOriginalTicket function'

3.6 The GoogleTranslateService

The 'GoogleTranslateService' is the communication between the support feature's view and the translate model. This service contains functions which call the respected 'GoogleTranslateModel functions'.

3.6.1 getModel

One of the first functions found in the 'GoogleTranslateService' is the 'getModel function'. This function is called in almost every function because it instantiates a 'GoogleTranslateModel' which can then be used to call functions inside the 'GoogleTranslateModel' file.

```
protected function getModel($clientid, $userid, $dbserver) {
    $out=new GoogleTranslateModel($clientid, $userid, $dbserver);
    return $out;
}
```

Figure 22: 'getModel function' inside the 'GoogleTranslateService'

3.6.2 translateOnPageLoad

This is the function that gets called whenever a support ticket is loaded. It calls the 'translateOnPageLoad function' in the 'GoogleTranslateModel' which results with either the translated or original ticket, based on the enabled settings of the current profile.

```
public function translateOnPageLoad($sid, $entity, $entityId, $ticket){
    $googleTranslateModel = $this->getGoogleTranslateModelInstance($sid);
    return $googleTranslateModel->translateOnPageLoad($entity, $entityId, $ticket);

}
```

Figure 23: 'translateOnPageLoad function' inside the 'GoogleTranslateService'

3.6.3 translateTicket

The 'translateTicket function' is the function that is called when the translate button is pressed. Inside this function, the' translateOnButtonPress function' of the 'GoogleTranslateModel' is called. This function returns a ticket and a boolean containing a true or false value. This boolean, 'isTranslated', indicates if the given ticket has been translated or not. If the ticket has not been translated, the original ticket is retrieved from the database.

The reason the original ticket is retrieved from the database instead of returning the given ticket is because the given ticket can already be translated. When an already translated ticket is used with the 'translateTicket function', the service calls the model with the given ticket as a parameter. The model, however, is not going to translate the given ticket because it is already written in the desired language, meaning that it returns the given ticket and the 'isTranslated boolean' with a false value.

If the service does not retrieve the original ticket afterwards, the already translated ticket is returned with 'isTranslated' set to false, meaning that the front end gets the wrong boolean value and the original ticket is never shown due to the fact that a translated ticket remains a translated ticket.

```
public function translateTicket($sid, $entity, $entityId, $ticket){

    $googleTranslateModel = $this->getGoogleTranslateModelInstance($sid);
    $ticket = $googleTranslateModel->translateOnButtonPress($entity, $entityId, $ticket);

    if($ticket["isTranslated"] == false){
        $sc = new SupportService();
        $origTicket = $sc->getDetail($sid, $ticket["ticket"]["uid"]);

        return ["isTranslated" => $ticket["isTranslated"],
            "ticket" => $origTicket];
    }
    return $ticket;
}
```

Figure 24: 'translateTicket function' inside the GoogleTranslateService

3.6.4 saveOriginalTicket

This is the function that is called just before a reply is added to the currently selected support ticket and either returns the original or translated ticket.

First, the setting value is retrieved from the model, being either true or false. When the value is true, it means that the given reply has to be translated to the original language of the selected ticket. If the value is false, it means that the reply can be added to the selected ticket in its original state.

When the value is true, an instance of the 'SupportService' is made. This is to retrieve the currently selected support ticket. With this ticket retrieved, the language of the first message is detected, being the original message that created the support ticket. This detected language is then used to translate the reply to the detected language by calling the 'translateSourceTextToGivenLanguageCode function'. This function returns the translated version of the provided reply which is then used to overwrite the original reply.

Lastly, the ticket is returned to the controller.

```
public function saveOriginalTicket($sid, $entity, $entityId, $ticket, $ticketData){

    if($this->saveOriginalEnabled($sid)){
        $sc = new SupportService();
        $originalTicket = $sc->getDetail($sid, $ticket["uid"]);

        $detectedLanguageCode = $this->detectLanguageCode($sid, $originalTicket["messages"][0]["message"]);
        $ticketData["vraag"] = $this->translateSourceTextToGivenLanguageCode($sid, $entity, $entityId, $ticketData["vraag"], $detectedLanguageCode);
    }
    return $ticketData;
}
```

Figure 25: 'saveOriginalTicket function' inside the GoogleTranslateService

3.6.5 Model calls

Besides the previously mentioned functions, separate functions are written to allow each model function to be used individually. This results in the service being reusable in a variety of scenarios as every function related to translating can be found in this service.

```
public function getCurrentLanguageCode($sid) {
    $googleTranslateModel = $this->getGoogleTranslateModelInstance($sid);
    return $googleTranslateModel->getCurrentLanguageCode();
}

public function detectLanguageCode($sid, $sourceText){
    $googleTranslateModel = $this->getGoogleTranslateModelInstance($sid);
    return $googleTranslateModel->googleTranslateDetectLanguageCode($sourceText);
}
```

Figure 26: Example functions of separate model function calls

3.7 The GoogleTranslateModel

This is the place which holds the most logic related to translating text and language detection. The most important functions of this model are investigated.

3.7.1 translateOnPageLoad

This function is called by the 'GoogleTranslateService' whenever a ticket is loaded. It receives 'entity, entityId and ticket' as parameters.

First, the translate setting status of the current profile is investigated. This, because when this setting is false, the translate function in general is disabled.

When the translate status is enabled, the automatic translate setting status is investigated. If this value is false, it means that the automatic translate setting is not enabled and the currently selected ticket should not be translated.

However, when this automatic translate setting status is enabled, the mode of the currently selected ticket is verified. If this mode is not "OFF", the 'entity, entityId and ticket' are forwarded to the translateTicket function, which translates the ticket to the current user's profile language.

When the mode is "OFF" or the general or auto translate settings are disabled, the 'translateOnPageLoad function' does not return a translated ticket and instead returns the original support ticket and the 'isTranslated boolean' with a false value.

```
public function translateOnPageLoad($entity, $entityId, $ticket){

    $isGoogleTranslateEnabled = $this->translateEnabled();
    if($isGoogleTranslateEnabled){

        $isAutoTranslateEnabled = $this->autoTranslateEnabled();
        if($isAutoTranslateEnabled){

            $translatingForSelectedTicketMode = $this->getGoogleTranslateMode($ticket["uid"]);
            if($translatingForSelectedTicketMode != self::OFF){
                $translatedTicket = $this->translateTicket($entity, $entityId, $ticket);
                return ["isTranslated" => true,
                    "ticket" => $translatedTicket];
            }
        }
    }

    return ["isTranslated" => false,
        "ticket" => $ticket];
}
```

Figure 27: 'translateOnPageLoad function'

3.7.2 translateOnButtonPress

This is the function that the service calls whenever the translate button is pressed. This function, looks at the user's settings first.

The first settings that is looked at is the general translate setting. When this setting is enabled, the automatic translate setting is investigated. This setting is quite important in the 'translateOnButtonPress function' as it indicates what happens when the button is pressed.

When the automatic translate setting is enabled, the mode of the currently selected ticket becomes relevant. When the translate button is pressed with this setting enabled, the 'switchGoogleTranslateMode function' is called which either creates a database entry of the currently selected ticket for the current user in the 'gt_translate_mode' table, or switches the mode of the already existing entry to the opposite value.

Next, when the mode is "ON", the ticket is translated. Afterwards, the translated ticket is compared with the original ticket to compare if the tickets are the same. If the tickets are the same, the translated ticket and the 'isTranslated boolean' with a false value are returned. If the tickets are not the same, the translated ticket and the 'isTranslated boolean' with a true value are returned.

When the automatic translate setting is disabled, the 'translateOnButtonPress function' translates the ticket and returns the 'translatedTicket' and an 'isTranslated boolean' based on if the ticket has been translated or not. However, in this scenario the boolean is only false when the translated ticket is the same as the original ticket, as this means that the ticket was already translated or the same language as the user's profile.

```
if($translatingForSelectedTicketMode == self::ON){

    $translatedTicket = $this->translateTicket($entity, $entityId, $ticket);
    if($translatedTicket == $ticket){
        return ["isTranslated" => false,
            "ticket" => $translatedTicket];
    } else{
        return ["isTranslated" => true,
            "ticket" => $translatedTicket];
    }
```

Figure 28: Code snippet of the translateOnButtonPress function

3.7.3 translateSourceTextToProfileLanguage

This is the function that is used whenever a ticket has to be translated and has 'entity, entityId and sourceText' as parameters.

First, a list of excluded languages is obtained from the 'getExcludedLanguages function'. After this, the language of the sourceText is detected with the 'googleTranslateDetectLanguageCode function'.

These two variables are then used to verify that the detected language is not one of the excluded languages. If the detected language is written in one of the excluded languages, the original string is returned. If not, the current user's profile language is obtained with the use of the 'getCurrentLanguageCode function'. This language code is then hashed with the source text to an MD5 hash.

Next, the generated hash is searched in the Elasticsearch index with the use of the 'elasticsearchSearch function' to verify if the translation has already been made. This search is then stored in the 'result variable'. If the 'result variable' has more than zero hits, it means that the translation has already been made and is stored in the variable that holds the result of the 'elasticsearchSearch function'. If the result variable does not contain any hits, the given string is translated to the current user's profile language, stored in the Elasticsearch index and then returned. This way, the translations are always returned correctly when being translated for the first time.

```
if(sizeof($result['hits']['hits']) == 0) {

    if($sourceTextLanguageCode != $currentProfileLanguageCode){
        $translation = $this->googleTranslateTranslate($sourceText, $currentProfileLanguageCode);
        $this->elasticsearchStore($sourceTextLanguageCode, $currentProfileLanguageCode, $entity, $entityId, $orig_hash, $translation);
        return $translation['text'];
    } else{
        return $sourceText;
    }
}
```

Figure 29: Code snippet of the 'translateSourceTextToProfileLanguage function'

3.8 The test process

At Otys, a task is not done when a developer says so. A task has to be tested and reviewed by multiple people first, creating a strict yet effective test process.

Whenever a developer resolves a task, the task is tested by Otys's Test Engineer. The Test Engineer verifies if the task works the way it is supposed to and gives feedback if necessary. When the tester has no more feedback to give, the task is ready for code reviewing.

The code reviewing is initiated by pushing the code to the stable branch. The code is then available on BitBucket, a code reviewing platform, where it is reviewed before it gets applied to the stable branch. The submitted code, is reviewed by at least two mid-level or senior developers. These developers review the code for suspicious or unnecessary code and provide feedback if necessary.

Once the code is reviewed, the code is merged to the stable branch. The task is assigned back to the task owner, who reviews it one last time. If the task owner approves the result, the task is closed.

3.9 Code review feedback

During the code reviewing process on the Google Translate task, a few minor changes are made to the 'GoogleTranslateModel' and the controllers.

GoogleTranslateModel

Inside the 'GoogleTranslateModel', the "ON" and "OFF" values that were used to check if the mode of the currently selected ticket were on or off, are changed from string values to global constants. This results in better long term code as using constants prevent the possibility of mistyping values.

```
const ON = "ON";
const OFF = "OFF";
```

Figure 30: Defining of global constants.

```
if($translatingForSelectedTicketMode != self::OFF){
    $translatedTicket = $this->translateTicket($entity, $entityId, $ticket);
    return ["isTranslated" => true,
        "ticket" => $translatedTicket];
}
```

Figure 31: Usage of the OFF constant

Next, 'sleep(1)' is removed from the translate functions. The reason it was initially written, was to avoid the returning of an empty translation. This was the case when storing a string in Elasticsearch because of the small delay it had. This resulted in the 'search function' returning an empty string because the translation was still being stored. The 'sleep function' solved this issue by having the application wait for a second, but was bad practice.

To solve this issue, the translated text that is being stored in Elasticsearch is returned instead of searching for the just stored translation separately in the Elasticsearch index. This made it possible to remove the 'sleep function' and prevent an unnecessary call to the Elasticsearch index.

```
if(sizeof($result['hits']['hits']) == 0) {

    if($sourceTextLanguageCode != $currentProfileLanguageCode){
        $translation = $this->googleTranslateTranslate($sourceText, $currentProfileLanguageCode);
        $this->elasticsearchStore($sourceTextLanguageCode, $currentProfileLanguageCode, $entity, $entityId, $orig_hash, $translation);
        return $translation['text'];
    } else{
        return $sourceText;
    }
}
```

Figure 32: Storing of a new translation in the Elasticsearch index

Front end

The first mistake that was made, was the use of '$rootScope'. This is a variable that should be used as little as possible due to it being available in every file in the project. To prevent the use of '$rootScope', a separate global value had to be created which could then be injected into the necessary controllers or services, creating a shared singleton.

```
.value('translations', {
    isTranslated: false
})
```

Figure 33: Global value 'translations'

Once this value is injected, it can then be used to manipulate the data inside which results in the value of 'isTranslated' being manipulatable anywhere in the file, which is exactly what is needed. This change prevents the polluting of the '$rootScope variable' and results in a more efficient solution.

```
.then(function(res){
    translations.isTranslated = res.isTranslated;
    return res.ticket;
});
```

Figure 34: Using of the injected 'translations' singleton

Lastly, the use of the global 'otysEntities class' is used instead of hardcoding the values of the '$entity and $entityId variable'. This is in general a better practice as it removes the possibility of human error and creates a dynamic way of receiving the required data.

```
var entityId = Entities.Support;
var entity = Entities.byId(entityId);
```

Figure 35: Usage of the 'otysEntities class'

4 Reflection

Implementing Google Translate into Otys's support feature has been a very interesting journey. I got into contact with quite a few new technologies such as Otys Ruksak, AngularJS, Google's Cloud Translation API and Elasticsearch. Each of these had their own learning curve which made this project very interesting.

This project has improved my debugging skills a lot which made me an overall better full-stack developer. This because, prior to this internship, my full-stack developing skills were lacking as I only focused on front end development during my IT-project. This resulted in the neglecting of my back end developing knowledge. However, during this internship I had to write so much more back end than front end code that it definitely closed the barrier between my front and back end knowledge which is going to be very handy in my future working career.

The knowledge I gained in the programming languages PHP and AngularJS is also going to be worth putting onto my resume, as these are very good languages to know. However, I am still more interested in learning newer technologies, instead of the older ones as I feel like these will be more useful in my future working career. Although, using older languages like PHP and AngularJS, both have its advantages and disadvantages but surely makes me appreciate the newer programming languages available.

Learning how to implement services like Google's Cloud Translation API is also something I am very happy about. This, because I am sure that there will be similar projects and or implementations in the future which are going to be twice as easy to implement because of the experience I gathered implementing it for Otys.

I also learned how to read and understand the documentation of APIs better, because of the experience I gathered implementing Google's Cloud Translation API and creating an Elasticsearch index. This, because at first I did not spend much attention to the documentation of these services, assuming I could implement it by myself without reading much. This was a big mistake and resulted in me wasting quite some time and eventually going back to the documentation to read it more clearly. Reading the documentation more clearly resulted in me being able to resolve my problems quite rapidly and learn the importance of reading and understanding the provided documentation.

This project was a very pleasant project to work on as it had its challenges and obstacles which kept the journey enjoyable and interesting. I am looking forward to working on projects like this in the future!

II. Research topic

1 Research question

Otys is an international company with four different international offices and over 1200 customers worldwide. Having so many different nationalities, comes with certain obstacles. One of those obstacles is the communication between the customers and the developer from Otys.

Inevitably, there will be customers that are unable to communicate with developers from Otys, which is a problem that needs to be solved. This is where Otys thought of implementing Google Translate. Implementing this will provide the possibility of automatically detecting languages and translating certain pages to the user's selected profile language which would quickly solve the previously indicated obstacle.

In this whole concept of online translating, Google Translate is definitely a popular name. However, is Google Translate the best option for Otys? This question is what is investigated in this research to conclude which option of online language detecting and translating is most interesting for Otys.

2 Research method

2.1 Approach

In this research, the competitors of Google Translate on the topics of language detecting and translating are looked at. Firstly, the function is investigated by looking into the code and logic behind it. This is done by looking at tutorials of how language translations and detections are made, which provides information of how this can be done on a larger scale.

Next, the options such as online APIs, packages and services for detecting and translating languages are explored, with a focus on variables like setup and usage documentation, translation and detection accuracy, and price per character.

Lastly, the advantages and disadvantages of each option are provided in a table, comparing the different options with each other. These results are reviewed and formulated into a conclusion, suiting the needs of Otys the best.

2.2 Analysis

To gather information, a multitude of sources like online articles about all kinds of language detecting and translating APIs and libraries are examined. The respected documentation and setup guides are consulted to get more information on the products and how it works.

Next, the APIs and libraries are tested on speed and accuracy, and the price of each option is consulted and reviewed with each other.

2.3 Comparison

All the advantages and disadvantages of each API and library are critically reviewed and compared. Using the gathered information makes it clear which product is best suited for which situation and helps conclude which service is best suited for Otys.

3 Research

3.1 Language Detection [1]

3.1.1 What is Language Detection

Natural Language Identification, is the problem of identifying which natural language a piece of content is written in.

3.1.2 How Language Detection works

To detect a language, there are two most commonly used techniques:

- Stop words technique
- N-grams technique

Stop words

The stop words technique is the simplest but effective approach for language classifications. For this technique, a corpus of words for each language has to be maintained, which is then used to check the given text on the number of occurrences of such words. The number of occurrences is then compared for each different language to identify the strongest correlation to a certain language. A very common strategy is to choose the most used words, which are usually stop words, and use these as a corpus to check which stop words are most frequently used in given text.

Lists of these stop words are very easily found on the internet and provide a good idea of how these words can be used to detect languages. These lists can be found of almost any language which makes this technique a simple alternative.

N-gram [1]

The N-grams technique is an advanced language detection algorithm which works by calculating and comparing language profiles of character N-gram sequences. This can be seen as an N-character slice of a string, with 'N' being dependant on the type of N-gram that is chosen.

An example with the word "TEXT":

- uni-grams: T, E, X, T
- bi-grams: _T, TE, EX, XT, T_
- tri-grams: __T, _TE, TEX, EXT, XT_, T__
- 4-grams: ___T, __TE, _TEX, TEXT, EXT_, XT__, T___

Figure 36: Example of different N-grams [1]

The most used range of 'N', varies from two to four. This is based on trade-offs which are depending on the size of the dataset, desired accuracy of the model, and desired speed and memory efficiency.

The whole n-gram technique has its benefits when it comes to errors and accuracy when provided less words. This, because when the text contains a few spelling errors, the remaining text is not affecting the other n-grams which would not be the case when using a technique that uses complete words, such as the stop word technique. Compared to the stop word technique, the n-gram technique requires less words to be accurate which makes this an even better technique to detect languages.

3.1.3 Where Language Detection is used

Natural language detection is used with neuro-linguistic programming (NLP) applications to detect the used language as accurately as possible. Examples of these are spam filtering, machine translation, etc...

Besides this, natural language detection is also used for search engines as they need to identify search queries and the language of each web page before the engine can decide whether to show the matched search results or not.

3.1.4 How Language Detection is checked

Checking the variety of options requires some consistent data. This is why there is a simple PHP file, as Otys is PHP focused, containing five different strings, each with a different length.

The PHP file contains a very short, short, medium, long and very long string which contain randomly generated text of the respected length in a certain language, meaning that each language has five different strings.

The languages that are tested are English, Dutch, French and Spanish, as Otys is mainly based in Europe and encounters these languages the most. These five strings are then used to detect the language using the different kinds of services. After each detection process, there is looked at the time it took to complete the request and how accurate the detection process was. This is done by looking at the amount of confidence the process returned.

When these five strings are detected, the five strings are then used in an array to detect all five of the languages at the same time. Here, only the duration of the request is looked at, as the detected languages and confidence is the same.

With these five requests done, an average of the amount of confidence and the duration of all the requests is made.

3.1.5 What are the options

3.1.5.1 Language Detection API [2]

Introduction

Language Detection API is an API with the main focus of detecting the language of a given text. It returns the detected language code with a confidence score and is currently detecting 164 languages.

It is recommended to use one of their official API clients made with Ruby, Python, Node, Java, PHP, Crystal or C#. However, it is also possible to use the API directly.

Usage

Following the setup guide provided, it took a total of twenty minutes to get the language detecting working in PHP. The reason it took this long was because of a small misconception in the API's readme file. In general, the installation process, using the PHP client, was well-written and easy to follow.

The client contains a 'detect function' which can be used to detect the language of one string. This function returns the language code, the confidence and if it is reliable. When the detected language code is reliable, the 'isReliable variable' contains a one. If not, it contains a zero.

The same 'detect function' can also be used to detect the language of multiple strings. To use this functionality, an array of strings is forwarded to the 'detect function'.

Technical

The Language Detection API PHP client v2.2.1, requires a PHP version of 5.3.0 or higher to work.

```
$time_start = microtime( get_as_float: true);

$results = DetectLanguage::detect($spanishveryshort);

$time_end = microtime( get_as_float: true);

echo  '<pre>'; print_r($results); echo '</pre>';

$execution_time = ($time_end - $time_start);

echo '<b>Total Execution Time:</b> '.$execution_time.' seconds';
```

Figure 37: Example of PHP code using the Language Detection API

Accuracy

English language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	26	En	9.86	0.1185870
Short text	285	En	12.18	0.1230969
Medium text	1285	En	11.30	0.1618218
Long text	3745	En	10.81	0.1768090
Very long text	15237	En	11.05	0.2146761
All of the above	20,578	/	/	0.2155555
		100%	Avg: 11.04	Avg: 0.1684243

Table 1: English text language detection with the Language Detection API

Dutch language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	30	Nl	15.52	0.1079351
Short text	372	Nl	8.1	0.1200931
Medium text	1030	Nl	8.22	0.1833770
Long text	3912	Nl	7.53	0.1685910
Very long text	14545	Nl	7.9	0.1783349
All of the above	19889	/	/	0.2568740
		100%	Avg: 9.45	Avg: 0.1692008

Table 2: Dutch text language detection with the Language Detection API

French language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	19	Fr	9.26	0.1111018
Short text	256	Fr	7.1	0.1322062
Medium text	1356	Fr	5.6	0.1724479
Long text	3540	Fr, En	5.03, 4.39	0.1616160
Very long text	15067	Fr	5.36	0.2266139
All of the above	20238	/	/	0.2343690
		100%	Avg: 6.47	Avg: 0.1730591

Table 3: French text language detection with the Language Detection API

Spanish language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	21	Sk	5.58	0.1493706
Short text	436	Es	8.8	0.1234459
Medium text	1239	Es	7.7	0.1785581
Long text	3975	Es	3.58	0.1604561
Very long text	15924	Es	3.58	0.2170610
All of the above	21595	/	/	0.2267148
		80%	Avg: 5.85	Avg: 0.1759344

Table 4: Spanish text language detection with the Language Detection API

Average

Name	Language detection accuracy	Average confidence	Average duration (seconds)
Language Detection API	95%	7.47	0.6866186

Table 5: Average results of the Language Detection API

Pricing

Plan name	Requests/day	MB/day	Price/month
Free	1000	1	Free
Basic	100,000	20	$5
Plus	1,000,000	200	$15
Premium	10,000,000	2000	$40

Table 6: Pricing of the Language Detection API

Overall review

Language Detection API is a fairly accurate but slow API when it comes to detecting the language of a string. Besides the speed, the costs can go all the way up to $40 a month which is quite a lot for a language detecting API compared to other options, like Landrok's language detector. The pricing system is also not the most optimal for Otys, as Otys sends out many small requests instead of fewer large requests. This means that a high limit of requests per day is needed which results in a more expensive monthly bill.

3.1.5.2 Patrick Schur's language detection package [3]

Introduction

Patrick Schur's language detection package is a free language detecting package which parses given training texts into n-gram sequences which it then uses to build a database file in JSON format to be used in the language detection phase. It can then take a given text and use the previously generated database to detect up to 110 languages.

Usage

Trying to get this package working took around 25 minutes as finding the language detection files in the project did not go very smoothly. A 'require_once' to the autoload.php file solved the issue.

This package provides a 'detect function' which can be used to detect a single string. On top of this 'detect function', a 'white or blacklist function' can be added to white or blacklist certain languages.

Besides the 'white and blacklist functions', a 'bestResult function' can be added to the 'detect function', only returning the best matched result. If the 'bestResult function' is too limited, a custom limit can be set by using the 'limit function' with the desired offset and length as parameters.

Technical

Patrick Schur's language detection package requires a PHP version of 7.0 or higher to work.

```php
$ld = new Language;

$time_start = microtime( get_as_float: true);

$result = $ld->detect($frenchverylong)->limit( offset: 0,  length: 3)->close();

echo '<pre>'; print_r($result); echo '</pre>';;

$time_end = microtime( get_as_float: true);

$execution_time = ($time_end - $time_start);

echo '<b>Total Execution Time:</b> '.$execution_time.' seconds';
```

Figure 38. Example of PHP code using Patrick Schur's language detection package

Accuracy

English language	Amount of chars	Detected language (Limited to 3)	Confidence (2 decimals)	Duration (seconds)
Very short text	26	En, La, Da	0.49, 0.41, 0.4	0.0050680
Short text	285	En, Da De	0.37, 0.29, 0.29	0.0072720
Medium text	1285	En, Fr, Af	0.48, 0.40, 0.39	0.0090141
Long text	3745	En, Ia, Fr	0.54, 0.42, 0.42	0.0121979
Very long text	15237	En, Ia, Fr	0.57, 0.45, 0.44	0.0247740
All of the above	20,578	/	/	0.0442869
		100%	Avg: 0.49 (En)	Avg: 0.0171021

Table 7: English text language detection with Patrick Schur's language detection package

Dutch language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	30	Nl, Af, Tr	0.49, 0.48, 0.43	0.0053009
Short text	372	Nl, Af, De	0.39, 0.35, 31	0.0075200
Medium text	1030	Nl, Af, De	0.47, 0.43, 0.40	0.0092220
Long text	3912	Nl, Af, De	0.59, 0.52, 0.48	0.0117630
Very long text	14545	Nl, Af, De	0.59, 0.52, 0.46	0.0235469
All of the above	19889	/	/	0.0450360
		100%	Avg: 0.59 (Nl)	Avg: 0.0170648

Table 8: Dutch text language detection with Patrick Schur's language detection package

French language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	19	Fr, Wo, Wa	0.41, 041, 0.38	0.0059640
Short text	256	Fr, Ia, En	0.35, 0.33, 0.31	0.0078361
Medium text	1356	Fr, Ca, Ia	0.48, 0.41, 0.39	0.0102820
Long text	3540	Fr, Ca, Ia	0.51, 0.43, 0.42	0.0117020
Very long text	15067	Fr, Ia, Ca	0.56, 0.44, 0.44	0.0234029
All of the above	20238	/	/	0.0461249
		100%	Avg: 0.56 (Fr)	Avg: 0.0175519

Table 9: French text language detection with Patrick Schur's language detection package

Spanish language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	21	Gl, Io, Pt-br	0.45, 0.44, 0.44	0.0062689
Short text	436	Es, Gl, Pt-br	0.35, 0.34, 0.33	0.0080890
Medium text	1239	Es, Io, Ia	0.45, 0.41, 0.41	0.0088942
Long text	3975	Es, Gl, Pt-br	0.54, 0.48, 0.47	0.0116548
Very long text	15924	Es, Ia, Gl	0.56, 0.52, 0.52	0.0247399
All of the above	21595	/	/	0.0457189
		100%*	Avg: 0,47 (Es*)	Avg: 0.0175609

Table 10: Spanish text language detection with Patrick Schur's language detection package

*Gl = Galician, is a language spoken in the north-western part of Spain where it is official along with Spanish [4]. As it can be considered a type of Spanish, the language detection is still classified as correct.

Average

Name	Language detection accuracy	Average confidence	Average duration (seconds)
Patrick Schur's language detection package	100%	0.53	0.0173199

Table 11: Average results of Patrick Schur's language detection package

Pricing

Plan name	Requests/day	MB/day	Price/month
None	Unlimited	Unlimited	Free

Table 12: Pricing of Patrick Schur's language detection package

Overall review

Patrick Schur's language detection package is a very reliable, fast and free option for detecting languages. It has a good chance of being the best option for Otys if the application is rewritten in a PHP version greater than or equal to 7.0. So unfortunately, this option is currently unavailable until Otys decides to upgrade to PHP 7.0 or newer.

3.1.5.3 Landrok's language detector [5]

Introduction

Landrok's language detector is a fast, reliable and databaseless language detecting library which is packaged with a 2MB dataset. It uses the n-gram algorithm and supports more than fifty languages.

Usage

Getting the language detector running was a very smooth and simple process. It took less than five minutes to get the language detector up and running as it only required a simple composer install.

The library has an 'evaluate function', which detects the language of the provided text. This function can only be used with a single string, which means no arrays. However, this can quickly be solved by creating a for each loop that evaluates every array item separately.

The 'evaluate function' only returns the language code of the language with the highest score. Although, a list of evaluated languages with the scores can be shown when using the 'getScores function'. This function does not support the possibility of choosing a custom score range, meaning that if fifty languages are evaluated, an array of fifty scores is returned.

Technical

Supported versions of PHP are 5.4, 5.5, 5.6, 7.0, 7.1, 7.2, 7.3 and HHVM

```php
$time_start = microtime( get_as_float: true);

$detector = new LanguageDetector\LanguageDetector();

$language = $detector->evaluate($englishLong)->getScores();

echo '<pre>'; print_r($language); echo '</pre>';;

$time_end = microtime( get_as_float: true);

$execution_time = ($time_end - $time_start);

echo "\nTotal Execution Time: ".$execution_time." seconds";
```

Figure 39: Example of PHP code using Landrok's language detector library

Accuracy

English language	Amount of chars	Detected language (Limited to 3)	Confidence (2 decimals)	Duration (seconds)
Very short text	26	En, Nl, Da	0.30, 0.27, 0.26	0.1171069
Short text	285	En, Nl, Da	0.65, 0.61, 0.60	0.1182370
Medium text	1285	En, Fr, Nl	0.81, 0.77, 0.76	0.2221648
Long text	3745	En, Nl, Af	0.89, 0.85, 0.83	0.2329368
Very long text	15237	En, Nl, Es	0.95, 0.89, 0.88	1.7598190
All of the above	20,578	/	/	1.8932940
		100%	Avg: 0.72	Avg: 0.7239264

Table 13: English text language detection with Landrok's language detector

Dutch language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	30	Nl, Af, Et	0.27, 0.27, 0.25	0.1196279
Short text	372	Nl, Af, Da	0.68, 0.67, 0.63	0.1242330
Medium text	1030	Nl, Af, De	0.79, 0.78, 0.73	0.1321249
Long text	3912	Nl, Af, De	0.90, 0.89, 0.83	0.2451438
Very long text	14545	Nl, Af, De	0.95, 0.94, 0.89	1.6162122
All of the above	19889	/	/	1.7958858
		100%	Avg: 0.72	Avg: 0.6722046

Table 14: Dutch text language detection with Landrok's language detector

French language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	19	Fr, Ro, It	0.21, 0.20, 0.20	0.1235361
Short text	256	Fr, It, Es	0.60, 0.58, 0.58	0.1278588
Medium text	1356	Fr, It, Es	0.79, 0.77, 0.76	0.1391110
Long text	3540	Fr, It, Es	0.86, 0.84, 0.83	0.2210631
Very long text	15067	Fr, It, Es	0.90, 0.89, 0.89	1.7205920
All of the above	20238	/	/	1.8609628
		100%	Avg: 0.67	Avg: 0.6988540

Table 15: French text language detection with Landrok's language detector

Spanish language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	21	It, Fr, Es	0.22, 0.22, 0.22	0.1219968
Short text	436	Es, Pt, It	0.70, 0.67, 0.66	0.1246540
Medium text	1239	Es, It, Pt	0.82, 0.79, 0.77	0.1337339
Long text	3975	Es, It, Pt	0.89, 0.87, 0.85	0.2495701
Very long text	15924	Es, It, Pt	0.94, 0.93, 0.89	2.0845348
All of the above	21595	/	/	2.2079789
		80%	Avg: 0.71	Avg: 0.8204114

Table 16: Spanish text language detection with Landrok's language detector

Average

Name	Language detection accuracy	Average confidence	Average duration (seconds)
Landrok's language detector	95%	0.70	0.7288498

Table 17: Average results of Landrok's language detector

Pricing

Plan name	Requests/day	MB/day	Price/month
None	Unlimited	Unlimited	Free

Table 18: Pricing of Landrok's language detector

Overall review

Landrok's language detector is a relatively accurate but slow library. It takes Landrok's language detector on average over forty times as long to detect the language of a string compared to Patrick Schur's language detection package. The good aspect of this library is that it is easy to get running and completely free.

3.1.5.4 Cloud Translation API

Introduction [6]

Cloud Translation is the most famous translating service. It detects more than 100 different languages, is very scalable, hosted by Google and is constantly being updated to create an even more accurate result.

Usage [6]

Getting the Cloud Translation API working does not take long at all. All it takes is a quick composer install and an API key to get it up and running. However, this API key requires credit card details as it is a paid API. Luckily, signing up results in a free 300$ worth of credit.

The Cloud Translation API has a few functions relevant to language detecting. One of them is the 'detectLanguage function'. This function returns the detected language of a given string. In case the language of multiple strings has to be detected, an array of strings can be provided to the 'detectLanguageBatch function'. This function then returns an array of all the detected languages. [7]

The API also provides a languages function which returns a list of all supported language codes.

Technical [8]

The Cloud Translation API can be used by C#, Go, Java, Node.js, PHP, Python and Ruby. The Google Cloud PHP Client that is used for this research is written in PHP version 7.2, but is also usable by older versions.

```php
$translateClient = new TranslateClient([
    'key' => $key,
]);

$time_start = microtime( get_as_float: true);

$result = $translateClient->detectLanguage($frenchverylong);

echo print_r($result);

$time_end = microtime( get_as_float: true);

$execution_time = ($time_end - $time_start);

echo 'Total Execution Time: '.$execution_time.' seconds'."\n";
```

Figure 40: Example of PHP code using the Cloud Translation API

Accuracy

English language	Amount of chars	Detected language (Limited to 3)	Confidence (2 decimals)	Duration (seconds)
Very short text	26	En	1	0.1426210
Short text	285	En	1	0.1755671
Medium text	1285	En	1	0.1473779
Long text	3745	En	1	0.2853870
Very long text	15237	En	1	0.3670690
All of the above	20,578	/	/	0.3781759
		100%	Avg: 1	Avg: 0.2018018

Table 19: English text language detection with the Cloud Translation API

Dutch language	**Amount of chars**	**Detected language**	**Confidence**	**Duration (seconds)**
Very short text	30	Nl	1	0.1649849
Short text	372	Nl	1	0.1494100
Medium text	1030	Nl	1	0.2443430
Long text	3912	Nl	1	0.2513930
Very long text	14545	Nl	1	0.3805489
All of the above	19889	/	/	0.4431231
		100%	Avg: 1	Avg: 0.2723004

Table 20: Dutch text language detection with the Cloud Translation API

French language	**Amount of chars**	**Detected language**	**Confidence**	**Duration (seconds)**
Very short text	19	Fr	1	0.1652860
Short text	256	Fr	0.92	0.1712210
Medium text	1356	Fr	0.86	0.2454109
Long text	3540	Fr	0.81	0.2602469
Very long text	15067	Fr	0.87	0.3798430
All of the above	20238	/	/	0.4151090
		100%	Avg: 0.89	Avg: 0.2743695

Table 21: French text language detection with the Cloud Translation API

Spanish language	**Amount of chars**	**Detected language**	**Confidence**	**Duration (seconds)**
Very short text	21	Es	1	0.1437361
Short text	436	Es	1	0.1396150
Medium text	1239	Es	1	0.1541130
Long text	3975	Es	1	0.2596190
Very long text	15924	Es	1	0.2995920
All of the above	21595	/	/	0.4376020
		100%	Avg: 1	Avg: 0.2390461

Table 22: Spanish text language detection with the Cloud Translation API

Average

Name	Language detection accuracy	Average confidence	Average duration (seconds)
Cloud Translation API	100%	0.97	0.2468794

Table 23: Average results of the Cloud Translation API

Pricing

Plan name	Pricing
Standard	$20 for every 1 milion character

Table 24: Pricing of the Cloud Translation API

Overall review

The Cloud Translation API is the current option that Otys uses. But is it that good compared to the other options? The Cloud Translation API is very accurate and fast compared to the Language Detection API. However, the price is an important factor here. Costing $20 for every million characters detected can be quite costly when used by many people. If money is irrelevant, this option is very good for Otys due to no PHP version limitation.

3.1.5.5 Pear's language detection package [9]

Introduction

Pear's language detection library is a php library created to identify human languages from text samples. Currently the library is maintained by inactive user Nicholas Pisarro.

Usage

Getting Pear's language detection library running did not take long at all. It required a quick pear install and a composer require to get the library working which was quite convenient. The library does not have any special functions besides a 'detectSimple and detect function'.

The 'detectSimple function' returns a string of the detected language, while the 'detect function' has an extra parameter which has the option to decide the desired amount of results. For example, when the value of three is provided, the three results with the highest confidence are returned.

Technical

Pear's Language Detection library requires a PHP version of 5.4 [10] or higher.

```php
$ld = new Text_LanguageDetect();
$ld->setNameMode( name_mode: 2);

$time_start = microtime( get_as_float: true);

$result = $ld->detect($englishVeryLong,  limit: 3);

$time_end = microtime( get_as_float: true);

echo print_r($result);

$execution_time = ($time_end - $time_start);
echo "\n".'Total Execution Time: '.$execution_time.' seconds'."\n";
```

Figure 41: Example of PHP code using the Pear Language Detection library

Accuracy

English language	Amount of chars	Detected language (Limited to 3)	Confidence (2 decimals)	Duration (seconds)
Very short text	26	En, Et, Da	0.30, 0.15, 0.13	0.0020950
Short text	285	En, [*], It	0.31, 0.20, 0.19	0.0033979
Medium text	1285	En, [*], De	0.36, 0.26, 0.24	0.0064900
Long text	3745	En, [*], Fr	0.45, 0.27, 0.24	0.0100350
Very long text	15237	En, [*], Fr	0.48, 0.30, 0.29	0.0228531
All of the above	20,578	/	/	0.0452180
		100%	Avg: 0.38	Avg: 0.0150148

Table 25: English text language detection with Pear's language detection package

*Empty bracket was returned. This is probably one of the two bugs that is unresolved.

Dutch language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	30	Nl, Et, Hu	0.28, 0.16, 0.16	0.0020640
Short text	372	Nl, Da, De	0.30, 0.22, 0.20	0.0038960
Medium text	1030	Nl, De, En	0.35, 0.25, 0.23	0.0059409
Long text	3912	Nl, De, Da	0.46, 0.30, 0.28	0.0095069
Very long text	14545	Nl, De, Da	0.50, 0.32, 0.27	0.0212709
All of the above	19889	/	/	0.0451269
		100%	Avg: 0.38	Avg: 0.0146343

Table 26: Dutch text language detection with Pear's language detection package

French language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	19	Ro, Fr, La	0.25, 0.23, 0.18	0.0020280
Short text	256	Fr, Es, It	0.26, 0.20, 0.19	0.0034740
Medium text	1356	Fr, Es, En	0.32, 0.23, 0.23	0.0079210
Long text	3540	Fr, Es, It	0.38, 0.26, 0.25	0.0097761
Very long text	15067	Fr, En, It	0.44, 0.29, 0.27	0.0218479
All of the above	20238	/	/	0.0414879
		80%	Avg: 0.33	Avg: 0.0144224

Table 27: French text language detection with Pear's language detection package

Spanish language	Amount of chars	Detected language	Confidence	Duration (seconds)
Very short text	21	Pt, Es, It	0.33, 0.25, 0.20	0.0019851
Short text	436	Es, It, Pt	0.23, 0.22, 0.20	0.0043011
Medium text	1239	Es, Pt, It	0.34, 0.28, 0.26	0.0067257
Long text	3975	Es, Pt, It	0.41, 0.36, 0.33	0.0122800
Very long text	15924	Es, Pt, It	0.44, 0.39, 0.34	0.0240300
All of the above	21595	/	/	0.0447490
		80%	Avg: 0.35	Avg: 0.0156785

Table 28: Spanish text language detection with Pear's language detection package

Average

Name	Language detection accuracy	Average confidence	Average duration (seconds)
Pear's Language Detection package	90%	0.36	0.0149375

Table 29: Average results of Pear's Language Detection package

Pricing

Plan name	Pricing
Standard	Free

Table 30: Pricing of Pear's Language Detection package

Overall review

Pear's Language Detection package is the fastest yet least accurate option, is completely free and easy to install. However, the loss in accuracy makes this option a bit less interesting compared to its competitors.

3.1.6 Comparison

The most important components of a language detection service, is its ability to detect the language fast and accurate for as little money as possible. Reviewing these five services provided a variety of results.

Name	Language detection accuracy	Average duration (seconds)	Price
Language Detection API	95%	0.6866186	$0 - $40*
Patrick Schur's language detection package	100%	0.0173199	Free
Landrok's language detector	95%	0.7288498	Free
Cloud Translation API	100%	0.2468794	$20/1M chars
Pear's Language Detection package	90%	0.0149375	Free

Table 31: Comparison of the different language detection options

*Dependant on plan

3.1.7 Conclusion

When looking at the results of each language detection option, the shorter strings is where the errors originate. Even the language detecting options that got the shorter strings right, did not have a lot of margin in confidence. This is where the importance of these errors in a real-life situation has to be looked at.

In a real-life scenario, a support ticket has a length of anywhere from 50 to 500 characters, which means that the length of the support ticket is relevant but only to a certain extent. This is why it is more important to look at the “short text” string compared to the “very short text” string, as this is a string of around 250 – 400 characters.

The “short text” string’s language, has been successfully detected by every language detection option. This means that in a real-life use case, the chance of all language detection options detecting the right language is very high. In result, it is more interesting to look at how fast a language was detected compared to how accurate it was.

The speed of which the language detection options detected the language, is on average for all options less than a second. Nonetheless, Patrick Schur’s language detection package and Pear’s Language Detection package have an average language detection of less than 0.02 seconds. However, Pear’s Language Detection package is a bit faster with an average language detecting duration of 0.0149375 seconds, compared to Patrick Schur’s 0.0173199 seconds.

Price-wise, the free options take the win as they can be used as many times a day as the user pleases. This results in Patrick Schur’s language detection package, Landrok’s language detector and Pear’s Language Detection package being the best options when it comes to the price.

In conclusion, it is safe to conclude that Patrick Schur’s language detection package is the best option out of all. Unfortunately, due to a restriction of PHP 7.0, this option can only be used after upgrading to PHP 7.0. This means that the question lies between Pear’s fast Language Detection package, which sacrifices a bit of accuracy at a free cost, versus the Cloud Translation API which has an almost perfect accuracy with a medium detection speed at a high cost of $20 per million characters.

Taking into account the average length of a support ticket and how accurate every option is when given a string of 250-400 character, the best option for Otys is Pear’s Language Detection package. This due to its fast, free and relatively accurate language detection. However, when Otys upgrades to PHP 7.0, Patrick Schur’s language detection package will be the best option.

3.2 Text Translation

3.2.1 What is Text Translation [11]

Machine Translation (MT) refers to automated software that has the possibility of translating source content into target languages. People can use MT to help them translate text into different languages.

3.2.2 Approaches [11]

There are three main approaches to machine translation:

Rule-based machine translation [12]:

Rule-based machine translation (RBMT) systems were the first commercial machine translation systems and are based on linguistic rules that allow the words to be put in different places and to have different meaning depending on the context.

Statistical systems [13]:

Statistical machine translation (SMT) analyses already existing human translations. SMT systems are phrased based, which means they take sequences of words, and creates translations with overlapping phrases. With this technology, it is possible to reduce the restrictions of word-based translations by translating full sequences of words with different lengths.

Neural MT [14]:

Neural machine translation (NMT) is a form of machine translation that uses a large artificial neural network to foresee the likelihood of a sequence of words, usually modelling sentences in an individually integrated model.

3.2.3 How Text Translation is checked

To check the variety of options available, a simple PHP file is used, as Otys is PHP focused, containing four different strings.

The four provided strings have a similar size but different complexity and are all written in different languages, Czech, Dutch, French and English. The reason these languages are chosen, is because these are the languages most spoken by the clients of Otys.

Complexities:

- Kid's tale: Contains limited and easily readable vocabulary.
- Fictional article: Contains mediocre and frequently used vocabulary and grammar.
- Scientific article: Contains sophisticated terminology and vocabulary.

Each text is translated to English and is afterwards reviewed for mistakes. Every major correction made to the translated text, is counted and evaluated, creating an overview of the number of mistakes every text translation option made. Besides the mistakes, the speed and price of the translations are also reviewed and compared with the other translate options.

All this data then provide a clear view of which option is best suited for Otys.

3.2.4 Used text

Kid's tale

Dutch version [15]

Een magere, sterke hond, die niet erg goed bekend stond, kwam eens op een dag langs een slagerswinkel. Daar zag hij een hoop smakelijke botten op de toonbank liggen. Hij pakte er een en rende weg.

Later, onderweg, passeerde hij een rivier. Halfweg de brug zag hij toevallig zijn spiegelbeeld in het water beneden zich. Omdat hij dacht, dat het een andere hond was, die een even smakelijk bot in zijn bek had, besloot hij handelend op te treden.

Hij gromde en hapte naar de hond in het water. Hij deed zijn kaken van elkaar om zijn scherpe tanden te laten zien en zo zijn vijand bang te maken. Natuurlijk viel het bot toen prompt uit zijn bek en plonsde het in het water. Het bot zonk naar de bodem buiten zijn bereik en het was voor altijd weg.

English version

A skinny, strong dog, who was not very well known, visited a butcher's shop one day. There, he saw a lot of tasty bones on the counter. He grabbed one and ran away.

Later, on the way, he passed a river. Halfway through the bridge he happened to see his reflection in the water below. Because he thought it was another dog that had an equally tasty bone in its mouth, he decided to act.

He growled and gasped at the dog in the water. He pulled his jaws apart to show his sharp teeth and scare his enemy. Of course the bone then fell out of his mouth promptly and splashed into the water. The bone sank to the bottom out of its reach and it was gone forever.

Czech version

Jeden hubený, silný pes, který nebyl příliš známý, navštívil jeden den řeznictví. Na pultu viděl spoustu chutných kostí. Popadl jednu a utekl.

Později na cestě míjel řeku. V polovině mostu viděl svůj odraz ve vodě . Protože si myslel, že je to další pes, který měl v ústech stejně chutnou kost, rozhodl se jednat.

Zavrčel a odfrkl na psa ve vodě. Vytáhl čelisti, aby ukázal své ostré zuby a vyděsil svého nepřítele. Ale upustil kost z úst a spadl do vody. kost se rychle potopila na dno za jeho dosah a byla navždy ztracena.

French version

Un chien maigre et fort, qui n'était pas très connu, s'est rendu un jour dans une boucherie. Là, il a vu beaucoup d'os savoureux sur le comptoir. Il en a attrapé un et s'est enfui.

Plus tard, sur le chemin, il a passé une rivière. Au milieu du pont, il voyait son reflet dans l'eau ci-dessous. Parce qu'il pensait que c'était un autre chien qui avait un os tout aussi savoureux dans la gueule, il décida d'agir.

Il grogna et haleta au chien dans l'eau. Il écarta les mâchoires pour montrer ses dents acérées et pour effrayer son ennemi. Bien sûr, l'os est ensuite rapidement tombé de sa bouche et s'est éclaboussé dans l'eau. L'os a coulé au fond hors de sa portée et il était parti pour toujours.

Fictional article

Dutch version

Afgelopen vrijdag is er om 9 uur 's avonds een kettingbotsing gebeurd op de snelweg richting Duitsland. Dit was ten oorzaken van hevige regen en lage wolken. Het aantal doden loopt al snel op tot vijf, waarvan drie Belgen en twee Duitsers.

De politie raadt de mensen af de snelweg richting Duitsland de komende 12 uur te gebruiken om zo files te vermijden.

English version

Last Friday at 9 o'clock in the evening a chain collision happened on the highway to Germany. This was due to heavy rain and low clouds. The number of deaths quickly rises to five, of which three are Belgian and two are German.

The police advise people not to use the highway to Germany for the next 12 hours to avoid traffic jams.

Czech version

Minulý pátek v 9 hodin večer se na dálnici do Německa stalo hromadná nehoda. Příčinou byl silný déšť a nízká oblačnost. Počet úmrtí rychle stoupá na pět, z nichž tři oběti jsou z Belgie a dvě z Německa .

Policie doporučuje lidem, aby po dalších 12 hodin nepoužívali dálnici do Německa, aby se vyhnuli obrovským dopravním zácpám.

French version

Vendredi dernier à 9 heures du soir, une collision en chaîne s'est produite sur l'autoroute en direction de l'Allemagne. Cela était dû aux fortes pluies et aux nuages bas. Le nombre de morts s'est élèvé rapidement à cinq, dont trois belges et deux allemands.

La police recommande aux personnes de ne pas emprunter l'autoroute en direction d'Allemagne pendant les 12 prochaines heures pour éviter les embouteillages.

Scientific article

Dutch version

In het verleden betekende dit een minicomputer en tegenwoordig een microprocessor, maar in beide gevallen leven de tradities van programmeren op machineniveau, die van nature de pioniersdagen van computing domineerden, verder in het real-time domein. Daarom is er een sterke affiniteit tussen de activiteiten van de samenleveing en die van de real-time computerwereld.

English version [16]

In the past this meant a minicomputer and nowadays a microprocessor, but in either case the traditions of machine level programming, that naturally dominated the pioneering days of computing, live on in the real-time domain. There is therefore a strong affinity between the activities of the Society and the real-time computing world.

Czech version

V minulosti to znamenalo minipočítač a v současné době mikroprocesor, ale v obou případech tradice strojového programování, která přirozeně dominovala průkopnickému období práce na počítači, přežívá až dodnes. Proto existuje silná spřízněnost mezi aktivitami Společnosti a současným počítačovým světem.

French version

Dans le passé, il s'agissait d'un mini-ordinateur et d'un microprocesseur, mais dans les deux cas, les traditions de la programmation au niveau de la machine, qui ont naturellement dominés les débuts de l'informatique, perdurent dans le domaine du temps réel. Il existe donc une forte affinité entre les activités de la société et celles du monde de l'informatique en temps réel.

3.2.5 What are the options

3.1.5.4 Cloud Translation API

Introduction [6]

Cloud Translation is the most famous translating service. It detects more than 100 different languages, is very scalable, hosted by Google is constantly being updated to create an even more accurate result.

Usage [17]

Getting Google's Cloud Translation API ready did not take any extra time as it was already installed for the language detection part of this research. However, getting the API ready is a very easy and straight forward process due to Google's simple setup guides found on the Google Cloud Platform.

Technical [18]

As mentioned above, the Cloud Translation API can be used by C#, Go, Java, Node.js, PHP, Python and Ruby. The Google Cloud PHP Client that is used for this research is written in PHP version 7.2, but is also usable by older versions.

```php
foreach ($dutchTextArray as $text) {
    $time_start = microtime( get_as_float: true);

    $result = $translateClient->translate($text, [
        'target' => $targetLang
    ]);

    $time_end = microtime( get_as_float: true);

    echo print_r($result['text']);

    $execution_time = ($time_end - $time_start);
    echo "\n".'Total Execution Time: '.$execution_time.' seconds'."\n";
}
```

Figure 42: Example of PHP code using the Cloud Translation API to translate an array of text

Accuracy

Dutch to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	0	0.4713380	No
Medium	0	0.0633981	No
Hard	0	0.1320028	No
	Total: 0	Avg: 0.2222463	0%

Table 32: Dutch to English text translation results of the Cloud Translation API

French to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	0	0.1463029	No
Medium	0	0.0697190	No
Hard	0	0.0593049	No
	Total: 0	Avg: 0.0917756	0%

Table 33: French to English text translation results of the Cloud Translation API

Czech to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	1	0.4462859	Yes
Medium	1	0.1819918	Yes
Hard	0	0.3706610	No
	Total: 2	Avg: 0.3329796	66.66%

Table 34: Czech to English text translation results of the Cloud Translation API

Average

Name	Total amount of critical corrections	Average duration (seconds)	Average misconception rate
Cloud Translation API	2	0.2156672	22.22%

Table 35: Average text translation results of the Cloud Translation API

Pricing

Plan name	Pricing
Standard	$20 for every 1 milion character

Table 36: Text translation pricing of the Cloud Translation API

Overall review

The Cloud Translation API is the current option that Otys uses. It provides very accurate and fast translations. However, the price of $20 for every million characters is what makes this option a bit less interesting. As mentioned in the language detection part of this research, if money is irrelevant, Google's Cloud Translation API would certainly be a good option.

3.1.5.4 Microsoft's Translator Text API

Introduction [19]

Microsoft, also being a known name in the information technology industry, also has its own text translating API. It is able to detect over sixty different languages and can be used with any operating system for text-to-text language translation. It uses Neural Machine Translating to provide high-quality AI-powered machine translations.

Usage [20]

Getting Microsoft's Translator Text API running is a bit more complicated than expected. At first, it requires an Azure account, which does not take that long to create. Next, an API key is needed. This can be quite confusing as Microsoft Azure has a lot of features. One of the setup guides found on the internet, indicated to create a "Cognitive service", which has an API key. However, this API key does not work for translating. Eventually, it turns out a separate resource has to be created in the Azure portal, specifically for the Translator API. With this resource created, a working API key is provided.

With this API key and a quick online search, a complete PHP file for Microsoft's Translator API is available. With the PHP file and key ready, the translating can begin.

The example code provided by Microsoft, only provided a 'Translate function' without any extra features besides the possibility of translating one string to multiple languages in one go. This is done by adding the desired languages to the 'params variable'.

Technical [19]

Using Microsoft's Translator Text API is usable with C#, Go, Java with a JDK of 7 or later, Node.js with a version of 8.12.x or later, PHP with a version of 5.6.x or later, Python version 2.7 or 3.x or Ruby 2.4. For each of these programming languages a code example is available of how to use it in an application.

```php
foreach ($frenchTextArray as $text) {
    $time_start = microtime( get_as_float: true);

    $requestBody = array (
        array (
            'Text' => $text,
        ),
    );

    $content = json_encode($requestBody);
    $result = Translate ($host, $path, $key, $params, $content);

    $time_end = microtime( get_as_float: true);

    $json = json_encode(json_decode($result), options: JSON_UNESCAPED_UNICODE | JSON_PRETTY_PRINT);
    echo $json;

    $execution_time = ($time_end - $time_start);
    echo "\n".'Total Execution Time: '.$execution_time.' seconds'."\n";
}
```

Figure 43: Example of PHP code using Micosoft's Translation Text API to translate an array of text

Accuracy

Dutch to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	1	0.9249131	Yes
Medium	1	0.3250720	Yes
Hard	0	0.4971371	No
	Total: 2	Avg: 0.5823741	66.66%

Table 37: Dutch to English text translation results of Microsoft's Translator Text API

French to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	0	0.7983501	No
Medium	0	0.2511041	No
Hard	0	0.3503010	No
	Total: 0	Avg: 0.4665850	0%

Table 38: French to English text translation results of Microsoft's Translator Text API

Czech to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	2	0.2155261	Yes
Medium	1	0.2457909	Yes
Hard	1	0.3307621	Yes
	Total: 4	Avg: 0.2640263	100%

Table 39: Czech to English text translation results of Microsoft's Translator Text API

Average

Name	Total amount of critical corrections	Average duration (seconds)	Average misconception rate
Microsoft's Translator Text API	6	0.4379618	55.55%

Table 40: Average text translation results of Microsoft's Translator Text API

Pricing

Plan name	Pricing
Instance	2M chars/month
Pay-as-you-go	$10/M chars
Volume discunt: S2	$2,055,001/month - $8.22/M chars
Volume discunt: S3	$6,000/month - $6/M chars
Volume discunt: S4	$45,00/month - $4.5/M chars

Table 41: Text translation pricing of Microsoft's Translator Text API

Overall review

Microsoft's Translator Text API did not give a great impression when setting up due to it's not that easy to navigate Microsoft Azure portal. As a first time user, navigating here is quite overwhelming and can lead to quite some time looking for the right service.

However, with the API key and test file set up, Microsoft's API did not perform as well as expected. Using the test file was a lot more complicated than its competitors. The API's results are decent but far from flawless, with an average misconception rate of 77.66%. This API is definitely not as accurate as its competitors. However, the pricing of Microsoft's Translator Text API is one of the cheaper alternatives, pricing at $10 for every million characters. However, with the misconception rate being so high, the lower price might not be low enough for this API to be worth it over its competitors.

3.1.5.4 Yandex Translate API

Introduction [21]

The Yandex Translate API is a universal text translation tool which uses Yandex's developed machine translation technology. The API is made to allow developers to integrate machine translations into their services, applications and websites, and offers support for more than ninety languages.

Usage

To use Yandex's Translate API, an API key needs to be obtained. This can be done by creating an account and navigating to the "API keys" tab. Here, a click on the "Create a new key" button generates the API key which can then be used to translate the desired fragments of text.

Next, a script is needed to use the API. For this, a quick search on the internet results in an unofficial Yandex PHP [22] package on GitHub. To implement this package, a few lines are added to the composer.json file, followed by a composer install. With this installed, the Yandex Translate API can be used.

The functions provided by this package are 'translate', 'getSource', 'getSourceLanguage' and 'getResultLanguage'. The 'translate function' has the functionality to translate the given bit of text to the provided language.

When the result of a translation is stored in a variable, the other three functions can be used on that variable. Each of these functions then return the result the function name implies.

Technical

Using this PHP package requires a PHP version of 5.3 or later. Besides PHP, the API can be used in an HTTPS format or with the use of CURL.

```php
$translator = new Translator($key);

foreach ($dutchTextArray as $text) {
    $time_start = microtime( get_as_float: true);
    $translation = $translator->translate($text, $targetLang);
    $time_end = microtime( get_as_float: true);

    echo print_r( expression: $translation . "\n");
    $execution_time = ($time_end - $time_start);
    echo "\n".'Total Execution Time: '.$execution_time.' seconds'."\n";
}
```

Figure 44: Example of PHP code using Yandex's Translate API to translate an array of text

Accuracy

Dutch to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	4	1.2659990	Yes
Medium	0	0.6874439	No
Hard	1	0.4621880	Yes
	Total: 5	0.8052103	66.66%

Table 42: Dutch to English text translation results of Yandex's Translate API

French to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	0	1.2240948	No
Medium	1	0.6035750	Yes
Hard	0	0.5134909	No
	Total: 1	0.7803869	0.33%

Table 43: French to English text translation results of Yandex's Translate API

Czech to English

Difficulty of text	Amount of critical corrections	Duration (seconds)	Do the mistakes result in misconceptions (Yes/No)
Easy	2	0.8393910	Yes
Medium	2	0.3288631	Yes
Hard	0	0.3024690	No
	Total: 4	0.4902410	66.66%

Table 44: Czech to English text translation results of Yandex's Translate API

Average

Name	Total amount of critical corrections	Average duration (seconds)	Average misconception rate
Yandex Translate	10	0.6919460	55.55%

Table 45: Average text translation results of Yandex's Translate API

Pricing

Plan name	Pricing
Free	1M/day and 10M/month
Number of characters for the reporting period	< 50M = $15/M char 50M < 100M = $12/M char 100M < 200M = $10/M char 200M < 500M = $8/M char 500M < 1000M = $6/M char

Table 46: Text translation pricing of Yandex's Translate API

Overall review

Using Yandex's Translate API is one of the most straight forward APIs to use. Getting an API key only takes three clicks and is then completely ready to use. Besides the API key, there are multiple clients created by Yandex users to simplify other people's experience which makes Yandex a very easy to use competitor.

However, the accuracy of Yandex is definitely not as good as its competitors, having a total of ten critical mistakes. Besides the critical mistakes, Yandex's Translate API makes quite some grammatical mistakes. These grammatical mistakes often make the text still understandable, but make it quite a bit more unclear for the user.

3.2.6 Comparison

The most important components of a language translation API, is its ability to detect the language fast and accurate for as little money as possible. Reviewing these five services provided a variety of results.

Name	**Total amount of critical corrections**	**Average misconception rate***	**Average duration (seconds)**	**Price**
Google's Cloud Translation API	2	22.22%	0.2156672	$20/M characters
Microsoft's Translator text API	6	55.55	0.4379618	1) 2M chars/month for free 2) $10/M chars
Yandex Translate API	10	55.55%	0.6919460	1) 1M/day and 10M/month for free 2) < 50M = $15/M char

Table 47: Comparison of the text translation options

*Chance of changing the context of a sentence which would create a misconception.

3.2.7 Conclusion

With the data provided from researching the text translating APIs, the best option can be concluded based on the speed, accuracy and pricing.

Google's Cloud Translation API is a very accurate and fast translating API, being the fastest and most accurate out of its competitors. It made three times less critical mistakes than Microsoft's Translator Text API and five times less critical mistakes than Yandex's Translate API, making this the best choice based on accuracy.

Speed wise, Google's Cloud Translation API is also a clear winner, with an average translating speed of around 0.2 seconds, being twice as fast as Microsoft's 0.4 second average and more than three times faster than Yandex's 0.7 second average.

However, pricing is where Google's Cloud Translation API cuts short, being the most expensive API out of its competitors with Microsoft's API being half of Google's price at $10 for every million characters compared to Google's $20 for every million characters. This is where Yandex is on the middle ground, pricing its service at $15 for every million characters translated.

With this provided data, Yandex is definitely not a valid alternative for Google's Cloud Translation API, being only $5 cheaper than Google with a drastic drop in accuracy and speed.

This results in Microsoft being left versus Google. Microsoft, being twice as cheap but also twice as slow and inaccurate, results in the discussion of price versus accuracy and speed. To solve this discussion, the actual use for Otys needs to be considered.

With text translation being used at the support feature of Otys, the amount of characters an average support ticket has and the importance of an accurate translation needs to be reviewed. With an average of 500 character for every support ticket, a small translating mistake can make a support ticket's core message become different and result in wrong information being delivered to the person reading. With this in mind, the importance of an accurate translation becomes even greater.

Besides the accuracy of translations, the amount of different tickets that can be translated with one million characters is around 2000. Having every translation stored in Elasticsearch, results in every ticket being translated only once. This means that not many API calls are made in general, as every translation is reused. This results in the pricing being a lot less important than the accuracy of the translation.

This information concludes that Google's Cloud Translation API is the best choice for translating Otys's support tickets.

4 Conclusion

To conclude this thesis, Otys's demand to use Google Translate in the Support feature of their application has been a success with new technologies being learned and personal skills being developed further. Skills like debugging, PHP programming and AngularJS, have all been developed and can be read more about in the reflection of my thesis task.

The research regarding Google Translate's competitors resulted in Google Translate being the best option for text translation but not the best for text detecting. Pear's and Patrick Schur's language detection packages are better and cheaper, making these two options the better choice in its own situation. More information about these two options can be found in the research part of this thesis.

During this thesis, I have developed my overall programming and professional skills a lot more. Not having to program in PHP for quite some time made back-end developing a bit harder at first. However, it did not stop me from completing my task and resulted in me further developing my back-end skills and becoming a better full-stack developer.

During the research part of this thesis, I learned about different packages and APIs. Due to the many packages and APIs I had to use in my research, I am able to read documentation more fluent and integrate different APIs and packages more efficiently.

Looking back at this bachelor project, I am proud of what I have achieved and how I worked during this whole internship period. Sometimes I had to wait for feedback and it seems that I did not have anything to do, but instead of doing nothing, I continued looking for things to work on, like looking for errors in my thesis, writing my blog or working on other tasks given by Otys.

I have become a lot more professional and will use all of the gathered knowledge and experience in my future working career.

5 Bibliography

[1] G. Singh, “Edit History for Natural Language Identification: What it is, why it is important, and how it works.,” CommonLounge, 2018. [Online]. Available: https://www.commonlounge.com/discussion/3ecabc3d82684d57a62ad8fbc200f43b/history.

[2] “Language Detection API,” Language Detection API, [Online]. Available: https://detectlanguage.com/.

[3] P. Schur, “GitHub - patrickschur/language-detection: A language detection library for PHP. Detects the language from a given text string.,” 2016 12 25. [Online]. Available: https://github.com/patrickschur/language-detection#basic-usage.

[4] “Galician language - Wikipedia,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Galician_language.

[5] Landrok, “GitHub - landrok/language-detector: A fast and reliable PHP library for detecting languages,” GitHub, 2 11 2016. [Online]. Available: https://github.com/landrok/language-detector/releases.

[6] Google, “Cloud Translation API - Dynamisch vertalen | Cloud Translation | Google Cloud,” [Online]. Available: https://cloud.google.com/translate/.

[7] Google, “google-cloud-php,” Google , [Online]. Available: https://googleapis.github.io/google-cloud-php/#/docs/google-cloud/v0.99.0/translate/translateclient.

[8] Google, “Detecting Language | Cloud Translation API | Google Cloud,” [Online]. Available: https://cloud.google.com/translate/docs/detecting-language.

[9] C. a. CloCkWeRX, “GitHub - pear/Text_LanguageDetect - PHP library to identify human languages from text samples.,” Pear, [Online]. Available: https://github.com/pear/Text_LanguageDetect.

[10] Pear, “Text_LanguageDetect,” Pear, [Online]. Available: https://pear.php.net/package/Text_LanguageDetect.

[11] Gala, “What is Machine Translation? | GALA Global,” GALA Global, [Online]. Available: https://www.gala-global.org/what-machine-translation.

[12] O. Technologies, “What is Rules Based Machine Translation - Omniscien Technologies,” Omniscien Technologies, [Online]. Available: https://omniscien.com/rules-based-machine-translation/.

[13] O. Technology, “What is Statistical Machine Translation (SMT) - Omniscien Technology,” Omniscien Technology, [Online]. Available: https://omniscien.com/?faqs=what-is-statistical-machine-translation-smt.

[14] “Neural Machine Translation - Wikipedia,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Neural_machine_translation.

[15] Aesopus, “De hond en het bot,” Volksverhalen Almanak, [Online]. Available: https://www.beleven.org/verhaal/de_hond_en_het_bot.

[16] A. Johnstone, “Computer resurrection issue 4,” Resurrection, Summer 1992. [Online]. Available: http://www.cs.man.ac.uk/CCS/res/res04.htm#g.

[17] Google, “Quickstart: Using Client Libraries | Cloud Translation API | Google Cloud,” Google, [Online]. Available: https://cloud.google.com/translate/docs/quickstart-client-libraries.

[18] Google, “Translating Text | Cloud Translation API | Google Cloud,” Google Cloud, [Online]. Available: https://cloud.google.com/translate/docs/translating-text.

[19] Microsoft, “Translator Text Documentation - Quickstart, Tutorials, API reference - Azure Cognitive Services | Microsoft Docs,” Microsoft Azure, [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/translator/.

[20] Microsoft, “Startpagina - Microsoft Azure,” Microsoft Azure, [Online]. Available: https://portal.azure.com/#home.

[21] Y. Translate, “Developers,” Yandex Translate, [Online]. Available: https://translate.yandex.com/developers.

[22] Nkt, “GitHub - yandex-php/translate-api: Client for Yandex.Translate API,” GitHub/Yandex, [Online]. Available: https://github.com/yandex-php/translate-api.

[23] Sebleier, “NLTK's list of English stopwords,” GitHub Gist, 08 2010. [Online]. Available: https://gist.github.com/sebleier/554280.