UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Dmytro Chasovskyi

Platform for a neural machine translation system demo and user data collection

Master's Thesis (30 ECTS)

Supervisor: Mark Fishel, PhD

Tartu 2017

Platvorm neuromasintõlke süsteemide demonstreerimiseks ning kasutaja andmete kogumiseks

Lühikokkuvõte:
Käesolev magistritöö keskendub probleemile, et ei ole olemas avaliku lähtekoodiga masintõlke platvormi, mis lubaks ka andmete kogumist lõpp-kasutajatelt. Sellise süsteemi olemasolu lihtsustaks hüpoteeside testimist ning eksperimentide läbiviimist nii gruppide kui ka individuaalsete uurijate poolt masintõlke valdkonnas. Lisaks sellele, saaks koguda ka tagasisidet lõppkasutajatelt. Töö autor arendas platvormi, valideeris ning täiustas seda pidevalt läbi kasutajate tagasiside. Autor on mõelnud ka platvormi arhitektuurilise disaini peale, kasutades tarkvaraarenduse parimaid tavasid. Selle tulemusena on olemas avatud lähtekoodiga platvorm masintõlke süsteemidele, mis võimaldab ka andmete kogumist.

Võtmesõnad: masintõlge, neuromasintõlge, crowd-sourcing ja kasutaja andmete kogumine

CERCS: P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Platform for a neural machine translation system demo and user data collection

Abstract:
The central problem this work focuses on is that there are no open-source platforms for machine translation systems, which also allow collecting data from end-users.

The existence of such a system will greatly simplify the testing of hypotheses and the conduction experiments both by groups and individual researchers in the field of machine translation as well a receiving feedback from end-users.

The author creates a platform from scratch, validates and continually improves its performance based on feedback from users. The author also thinks of the architecture design of the platform based on the best software engineering practices.

As a result, there is now an open-source platform for machine translation systems, which also allows collecting data from users.

Acknowledgements:
Firstly, I want to thank the Estonian Foreign Ministry's Development Cooperation and Humanitarian Aid funds, which supported my studies at the University of

Tartu and made my arrival in Estonia possible.

Secondly, I want to say thank you to my supervisor Mark Fishel and friends Ilya Verenich, Aqeel Labash, Aleksis Zalitis, Andreas Ellervee who helped me by providing suggestions and support during the period of thesis writing.

Thirdly, I want to say thank you to my family who has been with me in a hard time.

Fourthly, I would like to thank the Cloudware AS and its leaders Eivind Bøhn, Reidar Øksnevad, and Igor Orlov, as well as Marlon Dumas for their interest in getting me to the Master's degree.

Fifthly, I want to say thank you to everyone I have met during my studies in Estonia and did not have a chance to mention by names in previous gratitudes.

Finally, I want to say thank you to Anastasiia who believed in me from the beginning of our life intersection and continue believing in me, inspiring to new achievements and overcoming the unprecedented peaks.

Keywords: machine translation system, neural machine translation, setup machine translation system

CERCS: P170, Computer science, numerical analysis, systems, control

Contents

1 Introduction

It is known[1] that the volume of data increases exponentially with time. With the growing of data, grows a need of automated translating tools. Unfortunately, some of them do not work well, and some do not even function properly. The rapid development of the field of machine translation in recent years shows that there is a need to test models and research approaches and present them to the world. Several proprietary solutions exist such as Google Translate[2], Bing Translator[3], also local Baltic companies such as Tilde[4]. However, to the best of the author's knowledge, there are no open-source analogs.

This work aims at filling this gap by developing an openly accessible solution to a collection and show research developments for independent scientists and NLP groups worldwide.

1.1 Problems and Hypotheses

There are many different methods to evaluate a quality of translated sentences such as human[5], automatic[6] and crowdsourcing[7].

One of the evaluation ideas is to assess the translation quality in comparison with alternative translations receiving constant feedback from users. This method is useful when there are no reference sentences to which translation can be compared. Such an approach is based on crowdsourcing.

The thesis is aimed at solving the following problem:

- creating the open-source platform for demonstrating the work of a neural machine translation system with the ability to collect end-users feedback.

At the same time, the hypotheses, which we would like to test are:

- whether the end users need such a translation system
- how quickly can evaluation data be collected

[1] `https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/`. Last accessed at 2017-05-17

[2] `https://translate.google.com/`. Last accessed at 2017-05-17

[3] `https://www.bing.com/translator`. Last accessed at 2017-05-17

[4] `https://www.tilde.lv/`. Last accessed at 2017-05-17

[5] `https://en.wikipedia.org/wiki/Evaluation_of_machine_translation#Human_evaluation`. Last accessed at 2017-05-17

[6] `https://en.wikipedia.org/wiki/Evaluation_of_machine_translation#Automatic_evaluation`. Last accessed at 2017-05-17

[7] An example of crowdsourcing usage in machine translation is described in [GLI14].

1.2 Contribution

The author's primary contribution is the creation of an open-source platform for testing and show-casing neural machine translation systems, as well as collecting user data to improve the system. An additional contribution is analyzing the performance of the platform by collecting implicit and explicit user feedback and incorporating that into the development process.

1.3 Terminology

There are several concepts which have many names and need to be highlighted.

The website for neural machine translation has several URLs `neuralmt.ee`, `neurot~olge.ee`, `neurotolge.ee`, `masintolge.ut.ee` and in the thesis will be referred to as "the platform", "the website", "the project", "the tool".

The customer who ordered the development is the research group of natural language processing[8], at the Institute of Computer Science, University of Tartu. In the thesis, the NLP group as the client will be referred to as "the client", "the stakeholder", "the customer".

1.4 Layout of the thesis

Chapter 2 describes the background and original concepts on which the system was built.

Section 3 goes through the system architecture and provides a detailed explanation of requirements, design, and implementation.

Chapter 4 analyses the current users of the system, the engine of growth and shows the aptitude of the system for real people.

Chapter 5 sums up the project and shortly repeats the findings from previous chapters.

Finally, chapter 6 outlines future work and possible development directions of the project.

[8] `http://nlp.cs.ut.ee/`. Last accessed at 2017-05-17

2 Background

This section makes a brief guide to the theoretical aspects of machine translation, open-source projects, and crowd-sourcing, the main fundamentals on which the thesis is based.

2.1 Machine translation

Machine translation (MT) is a branch of computational linguistics which is oriented on translation between different languages, usually as text or speech.

There are several different approaches to MT such as rule-based translation, Statistical MT, and Neural MT.

Statistical machine translation (SMT) [BPPM93] is based on the idea that each word can be translated from one language to another. Therefore, based on corpora[9] we can make a frequency table and calculate a probability of a word from a source language corresponding to a word from a target language. Different words can be translated differently, so to use the power of context phrase-based models exist, first described in [KOM03].

Neural machine translation (NMT) [BCB14] is a relatively new approach to machine translation. It is based on the idea of having two Recurrent neural networks (RNNs)[10] where the first one RNN is the encoder, and the second one is the decoder. Simply speaking, a user[11] presents the system with source sentences and target sentences and the model tunes hidden weight vectors in the RNNs to ensure that the system output resembles the target sentences.

Rule-based translations are the oldest approach compared to NMT and SMT, approximately developed at the 1950s, which still has it application field but very restricted. There is a source sentence, it is parsed and analyzed, and after that, each word is translated and composed to an output sentence, using morphology and sometimes semantic analyses. A modern example of a rule-based system is the Apertium project[FGRN+11].

2.2 Open-source

Open-source means to have source code publicly available, everyone can contribute to the development and reuse it for his or her purposes unless it is restricted by a license. A good definition can be found at [TECb]. The author searched for

[9] `https://en.wikipedia.org/wiki/Corpora_in_Translation_Studies`. Last accessed at 2017-05-17

[10] `https://en.wikipedia.org/wiki/Recurrent_neural_network`. Last accessed at 2017-05-17

[11] By user rather a researcher means than a random person.

available solutions to setup machine translation system. Currently, there are no any open-source projects to make a website for such a system. Therefore, the author had to create the in-house solution. The main reason why to go open-source was to help machine translation research groups around the world host their websites and collect data through crowd-sourcing.

To make a project open-source much work is required, so to follow the most appropriate approach the author consulted with [Zak]. The article suggests making the following steps to make a project successful: create wiki pages, make the entrance level for contributors as low as possible, start by creating a community of end users, organize a mailing list.

The project is hosted on Github `https://github.com/ChameleonTartu/neurotolge`. In the current phase of the project, it requires much work to make it truly open-source and attractive for people for long-term contributions. At the same time, many milestones are already passed. For instance, the author created a readme document, an installation guide, and wiki pages.

There are several software licenses to choose from, to understand which options can be selected, [Bus] proposes good summing up and general overview on licenses. The author decided to move with the MIT license which is one the most widely used for individual projects[12].

2.3 Crowd-sourcing

Crowd-sourcing is a method of distributing tasks among people to solve small parts of it and combining in a big, usually complex task. Another definition is that many people do the same job or similar jobs and statistically make likely correct decisions more often. There are many examples in different spheres of life[13].

In machine translation, the usefulness of crowdsourcing is shown for example in [BBCB+13], where Amazon Mechanical Turk (Amazon MTurk) was used on a big scale, which helped to collect a significant amount of data. The main downside with Amazon MTurk is that it costs money which is an exclusively administrative cost. Therefore, at the same conference three years later this service was not in use, instead of this, work was done by research teams [BCF+16]. Nevertheless, since crowd-sourcing has shown excellent performance in the past, the thesis renovates the crowd-sourcing idea for NLP research groups worldwide.

[12] Based on `https://www.whitesourcesoftware.com/whitesource-blog/open-source-software-licenses-trends/`. Last accessed at 2017-05-16

[13] `https://en.wikipedia.org/wiki/Crowdsourcing`. Last accessed at 2017-05-17

3 System design

This section describes the general system structure of the platform, implementation cycle, and decisions behind the scene. The back-end section also includes additional description of integrations with external systems (e.g. Google, Tilde, the UT translator).

3.1 Requirements

While building the website, the author has followed several primary requirements:

- easy to collect and extract data
- reliable and can fetch several translations from different translators in a reasonable time frame
- easy to use and aesthetically pleasing
- easy to maintain
- open-source under the MIT license
- supports a vast majority of browsers
- possible to use from different devices such as mobile phones, tablets, and desktops
- possible to use simultaneously by many users

3.2 System architecture

This section will go in depth into current project architecture and will explain design choices made during development.

The first question was if the project should separate back-end and the front-end flows or should a template engine rather be used and generate all front-end from back-end? Several articles support and argue between these design choices. For instance, [Joh], [Car] and [Use] provide arguments why separation is beneficial: easier to develop, easier to change parts, more straightforward for component teams [Inn] in the sense of development. [Hus] excellently summarizes with clear arguments why it might be inconvenient to have tightly coupled front-end and back-end, mostly based on the same reasons as previous articles, but in more structured way.

Without any MVC frameworks for the front-end (See section 3.3.3) I decided to use the combination of Flask and Jinja2. The decision was based on the fact

that I was a single developer in a team and separation might delay the delivery of the project.

Jinja2 is a template engine which the author has used with Flask. The author relied on documentation which provided reasons to believe that this framework was one of the fastest to use[Ron].

The whole system is drawn on figure 1.

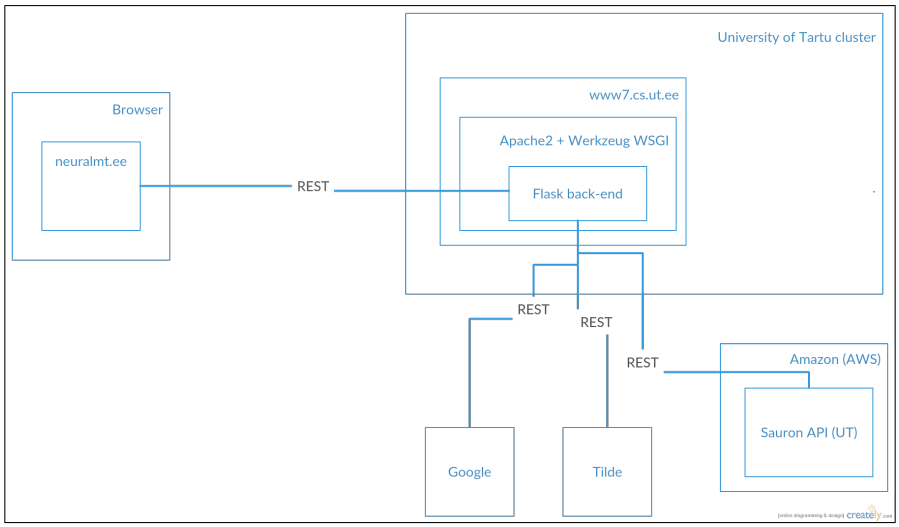


Figure 1: System architecture schema

3.3 Implementation

In this section, the author explains in a detailed manner the choices made while choosing the frameworks, the drawbacks, and the mistakes made through the application process. The decisions and the solutions which were created during the development cycle. Also, the author provides arguments for all steps and his point of view on each part of the project.

The project consists of several parts:

- back-end (See chapter 3.3.2)
- front-end (See section 3.3.3)
- infrastructure (See section 3.3.4)

Briefly, the back-end is implemented with Flask and Jinja2. The front-end implementation is done with jQuery, AJAX, simple JavaScript components. Infrastructure is based on Apache2 and Werkzeug (WSGI utility library for Python).

3.3.1 Development cycle

Nowadays, there are several methodologies for project management which have a lot of successful stories of being applied such as Scrum, Waterfall and Spiral. The author was a one-developer team. Therefore, not all state-of-the-art methods fitted well for the work. The author has stopped with the stakeholder on an in-between variant scrum and spiral model where the author implemented part of the functionality and got constant feedback weekly or biweekly depending on the client's availability. The project required agility because of constant changes after feedback and at the same time we iteratively completed each part of the project: front-end, back-end, infrastructure.

The cycle of iterations was the following: the author started with basic HTML, CSS prototyping and positioning of the web page. After that, the author has moved to the web page interaction features based on the feedback given by the client. Also, to simplify communication, we have a shared GitHub project as was mentioned before.

After finishing the front-end, part the author moved to the back-end. The first model was too complicated (instead of making GET requests the system did POST plus a state machine for applications which was over-engineered in the sense that one URL-handler could handle almost the whole system), after that the author moved to a more appropriate RESTful style in the project. The first deploy on the University of Tartu High-Performance Computing Cluster (HPC cluster) was complicated, because all manuals initially tried, were too outdated and required much work to find the one which led to the desired result, so the author deployed to Heroku[14]. After several weeks the author successfully moved back-end to the UT HPC cluster.

While going through iterations, the initial statistical machine translation system which the author integrated with the website was replaced by him with a neural machine translation system from the UT and reintegrated. After the project had gone alive, changes were made only to meet the designer requirements and modify the front-end.

3.3.2 Back-end

While choosing a framework for building the web application the author was guided by the following requirements for the language - framework pair:

- development should be fast (e.g. C++ will not satisfy this requirement)
- language should be as concise and as expressive as possible (e.g. modern JavaScript, Ruby, Python)

[14] `https://www.heroku.com/about`. Last accessed at 2017-05-13

Based on the described requirements the author has stopped on Ruby and Python.

Gareth Dwyer in his article [Dwy] claims that Flask is better for prototyping and gives more control over functionality included. In the meantime, it does not follow principle "batteries included" comparing to Django.

Another blog post on Slant[15] [com] suggests that Ruby on Rails (RoR) has more power and is easy to begin from, but at the same time is quite difficult if not following the established conventions, plus has a lot of them. Flask looks less attractive with a lack of advanced features, huge community, and enterprise success stories, but at the same time seems more friendly to open-minded developers who want to get their hands dirty.

Quora's[16] question [Ano] shows that community supports RoR more than any other framework, but at the same time idea of easy switching between frameworks such as Flask to Pyramid seems interesting. RoR from this angle has a huge disadvantage that there are no similar framework in Ruby to switch to in case RoR will be to complex, Sinatra, from my personal point of view, does not count, because it is too small and quite doubtfully can be used in production even it is from the same micro-framework family as Flask.

Based on the information the author has provided, the author has decided to go ahead with Flask, his primary objectives were:

- easy to use
- has a web-developer toolbar (Comparing to RoR)
- cross-platform in the sense of operating system server (in the time the author has been developing the application, the author was not sure which servers with which operation systems will be in use)
- good performance
- multi-threaded

There is a drawback that Flask does not support regular asynchronous requests. By the time the author made the decision to move with Flask, the author has underestimated the importance of the feature and how much trouble it may cause in the future.

Here is the main functionality of the back-end:

[15] `https://medium.com/building-slant/what-is-slant-5a836b200c0`. Last accessed at 2017-05-13

[16] `https://www.quora.com/about`. Last accessed at 2017-05-13

- request translation from Estonian to English or from English to Estonian by the UT translator (so-called "translate" mode)
- request translation with the play option from Estonian to English or from English to Estonian by Tilde, Google and the UT translators (aka "play" mode)
- get "main", "about" or "what's wrong" pages in different languages (Estonian, English)
- send the best translator, if any, chosen from Tilde, Google, the UT translators in the "play" mode

The back-end is built in a way that a single file contains all information about routing and responses (masintolge.py), and there are several self-written libraries which handle translators and include integrations with them.

Two main features of the back-end which I will describe in more details are "logging" and "parallel translations and collecting of results".

"Logging" is printing out events to error logs. By events, I mean everything that is happening during the back-end operating. It helps to track time for translations in batch and separately for each. It helps reproduce errors. For instance, in early stages of the project, Estonian symbols with diacritics were not handled correctly. By simple scripting statistics about translators were extracted from logs. The recording file is backed up automatically each week by the HPC cluster team.

"Parallel translations and collecting of results" is a feature to send parallel requests to translators (e.g. currently Tilde, Google, and the UT, previously it was Bing, Google, and the UT) and collect data into the queue, and return aggregated translations as a big batch to a user. As an implementation of this feature, the author has used Python FIFO Queue[Fou] with a size of three to synchronize all translations in a single place before sending to users. To make the application reliable all translator requests implement the save translate function which contains an exception handling wrapper. All translators were integrated through REST APIs.

To understand the queue expediency, a few calculations had to be done. Assuming that communication time between server and client is t and each translator has it is own speed. Google has t_{google}, Tilde - t_{tilde} and UT - t_{ut}. There are several analogical implementations that can be proposed:

- sequential back-end translation
- parallel back-end translation
- sequential front-end translation

- parallel front-end translation

Calculation for sequential back-end translation shows that total translation time will be approximately $2t + t_{ut} + t_{google} + t_{tilde}$.

Parallel back-end translation time will be $2t + \max(t_{ut}, t_{google}, t_{tilde})$.

Sequential front-end translation time will be $t_{ut} + t_{google} + t_{tilde}$.

Parallel front-end translation time will be $\max(t_{ut}, t_{google}, t_{tilde})$.

The fastest options are parallel front-end or back-end translations. The front-end option was rejected based on the following considerations:

- the website contains personal data, and in the back-end, it will be easier to protect credentials
- difference between these two methods are in milliseconds so that they can be counted as equally good
- doing the task in front-end is more complex than doing it in back-end

To summarize, the mathematical formulas prove expediency of Python FIFO Queue usage.

The project REST API uses JSON responses. List of all available requests with some examples below.

"Translate" mode:

```
GET /translate?from=<translate_from>&to=<translate_to>&q=<text>

Request: http://neuralmt.ee/translate?from=et&to=en&q=Tere

Response:

{
    "translations":
    [
        {
            "translator": "ut",
            "translation": "Hi"
        }
    ],
    "success": true
}
```

"Play" mode:

```
GET /play?from=<translate_from>&to=<translate_to>&q=<text>

Request: http://neurlamt.ee/play?from=et&to=en&q=Tere
```

```
Response:

{
    "translations":
    [
        {
            "translator": "google",
            "translation": "Hello"
        },
        {
            "translator": "tilde",
            "translation": "Welcome"
        },
        {
            "translator": "ut",
            "translation": "Hi"
        }
    ],
    "success": true
}

Request: http://neuralmt.ee/play?from=du&to=en&q=Hallo

Response:

{
    "success": false
}
```

Best translation choice:

```
POST /

{
    "google": <google_translation>
    "tilde": <tilde_translation>
    "ut": <ut_translation>
    "best_translator": <translator>
}

Response:
HTTP/1.1 201 CREATED
```

Get the main page with a specification of a language; default language is Estonian:

```
GET /et    Estonian version
GET /en    English version
GET /    the same as GET /et
GET /<arbitrary string>    The same as GET /et
```

Get about page:

```
GET /about/et    Estonian version
GET /about/en    English version
GET /about/<arbitrary string>   the same as GET /about/et (Exception
    GET /about/ which returns status 404 Page not found)
```

Get error page:

```
GET /error/et    Estonian version
GET /error/en    English version
GET /error/<arbitrary string>   the same as GET /about/et (Exception
    GET /error/ which returns status 404 Page not found)
```

There are several examples of requests to an external APIs integrated into the project. The integrations include sensitive information, so examples have undefined parameters.

Google requests:

```
GET https://translation.googleapis.com/language/translate/v2?key=<
    api_key>&source=<translate_from>&target=<translate_to>&q=<text>

Response for correct request:

{
    "data":
    {
        "translations":
        [
            {
                "translatedText": <translated_text>
            }
        ]
    }
}

Response for a request with wrong <language_from> parameter:

{
    "error":
    {
        "code": 400,
        "message": "Invalid Value",
        "errors":
        [
            {
                "message": "Invalid Value",
                "domain": "global",
                "reason": "invalid"
            }
```

```
        ]
    }
}
```

Tilde requests:

```
GET https://www.letsmt.eu/ws/service.svc/json/Translate?appID=<app_id
    >&systemID=<system_id>&options=%20HTTP/1.1&text=<text>
Header: client id: <client id>

Response with correct client id:

<translated_text>

Response with wrong client id:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.
    w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso
    -8859-1"/>
<title>401 - Unauthorized: Access is denied due to invalid
    credentials.</title>
<style type="text/css">
 <!
body{margin:0;font-size:.7em;font-family:Verdana, Arial, Helvetica,
    sans-serif;background:#EEEEEE;}
fieldset{padding:0 15px 10px 15px;}
h1{font-size:2.4em;margin:0;color:#FFF;}
h2{font-size:1.7em;margin:0;color:#CC0000;}
h3{font-size:1.2em;margin:10px 0 0 0;color:#000000;}
#header{width:96%;margin:0 0 0 0;padding:6px 2% 6px 2%;font-family:"
    trebuchet MS", Verdana, sans-serif;color:#FFF;
background-color:#555555;}
#content{margin:0 0 0 2%;position:relative;}
.content-container{background:#FFF;width:96%;margin-top:8px;padding
    :10px;position:relative;}
  >
</style>
</head>
<body>
<div id="header"><h1>Server Error</h1></div>
<div id="content">
 <div class="content-container"><fieldset>
  <h2>401 - Unauthorized: Access is denied due to invalid credentials
      .</h2>
  <h3>You do not have permission to view this directory or page using
       the credentials that you supplied.</h3>
 </fieldset></div>
```

```
</div>
</body>
</html>
```

The UT requests:

```
GTE https://urgas.ee/sauron/rest/translate/<language_pair>?auth=<auth
    token>&src=<text>

Response with correct authentication token:

{
    "tgt": <translated_text>
}

Response with illegal authentication token:

{
    "timestamp": 1494627948722,
    "status": 400,
    "error": "Bad Request",
    "exception": "ee.ut.sauron.exception.IllegalRequestException",
    "message":"Authentication failed",
    "path":"/sauron/rest/translate/<language_pair>"
}
```

To summarize this chapter, the author thinks that Flask was a good, but not an ideal choice for this project, and if in the future the author will start a similar project he will consider a much broader range of frameworks.

3.3.3 Front-end

The front-end is built with Bootstrap3, jQuery (v3.1.1), Javascript(ES5[Inta] and elements of ES6[Intb]), HTML5 and CSS3.

The author has decided to choose a framework to operate with components on the web page. Primary candidates were Bootstrap3 and Foundation. Features according to which the author has chosen Bootstrap3 were: supports IE8, supports LESS (which was supposed to be used but never was), and it has a huge community compared to Foundation. Foundation, of course, had its advantages and Bootstrap3 required many customizations, to look comparable to the initial design.

The author reviewed ten alternatives to Bootstrap and Foundation[Agu]. Quickly looking through frameworks, the author rejected them because of unnecessary complexity or because too often they do not have required components, which are too demanding to build from scratch with no strong prior experience in front-end engineering.

Therefore, in the end, the author has stayed with Bootstrap3. After that the next question was raised: whether the project needs any MVC framework such as AngularJS, ReactJS, KnockoutJS or it is enough to write small plug-ins to add simple interaction to a web page with JavaScript and jQuery? One article provides a comparison between ReactJS and AngularJS [Kor]. The article mentions that ReactJS is immature and even if it has a low learning curve it may be overwhelming for ReactJS newcomers. AngularJS is a bit better and has quite a big community and many tutorials, but at the same time has a higher learning curve for the first version while the second version does not have enough materials and is immature in the second version. At the same time [Har] provides a comparison between ReactJS, AngularJS, KnockoutJS and pure DOM manipulation. Even ReactJS shows the best results on average among all browsers. After serious considerations, the author decided to fully reject AngularJS because of the learning curve steep and immaturity of second version. In the meantime, pondering a lot about ReactJS, the author has chosen to move with a simpler solution, potentially, moving to ReactJS.

List of plug-ins which were implemented with JavaScript and jQuery:

- counting the already typed symbols in the main text area (includes the fact that Estonian letters with diacritics encoded as two bytes)
- sending asynchronous queries to server-side (with AJAX)
- rendering DOM (e.g. in "translate" and "play" modes)
- browser computations (e.g. shuffling of returned translations from back-end)
- changing languages for translations "language from", auto-update "language to"

More detailed explanation for each of the front-end features is below.

Sending asynchronous queries to a server-side using AJAX is explained in detail in the article [Teca]. AJAX is used to send the best translation with no interruption from the natural flow. There is a superior to AJAX technology web-sockets described at [Kon]. The main disadvantage is that Internet Explorer browsers do not support web-sockets before version 11[Dev]. Supporting IE is crucial because the project has to support as many browsers as possible.

Rendering DOM is the center part of the front-end, it updates the page depending on requests received and actions performed by a user. Rendering includes updating the footer, drawing images, showing and hiding translations, highlighting the best version of the translation, showing and hiding texts.

Browser computations are the shuffling of translations, filtering out empty translations and conditional rendering (e.g. rendering differently dependently on how many translations are available).

All these can be found in the GitHub. One feature that might require additional explanation is the language choice. The assumption was that not all languages might have an opposite pair. For example, French-English may exist and at the same time, English-French may not. Therefore, if the user chooses a language in "translate from" list, the automatically "translate to" list will be updated. For example, existing language pairs are French-Estonian, Estonian-English, English-Estonian, Estonian-Finish. If you choose Estonian language in the "translate from" drop-down list in "translate to" list only English will occur, at the same time if you choose French in "translate from" drop-down list only Estonian language in "translate to" list will occur. All these time "translate from" will contain French, English and Estonian languages. Such a decision was made by the author based on the feedback from early adopters to reduce confusion for users. Therefore, there will not be a situation when a user tries to use non-existing pairs like French-Finish.

Summarizing the chapter, not all decisions were ideal, but they were partially derived from constraints, partially, based on the author's preferences. In the long-term perspective, the author think that ReactJS should be used and back-end and front-end should be decoupled.

3.3.4 Infrastructure

Initially, the author hosted the project on Heroku, but it caused a lot of technical difficulties and made project dependent on a third party provider. To overcome this issue, the author did the UT HPC server migration.

Setup in the UT HPC cluster allowed making the system more reliable and controllable. Moreover, it allowed making fixes faster because of direct contact with HPC center (in case servers down or slow in response). There is no official visual statistics on how many accidents happened and the duration of each accident, but there was only one accident which lasted an hour and a half (the most noticeable on 4th May) from the words of an administrator. Otherwise, several smaller accidents has happened due to an unstable node in the cluster, which lasted no more than half an hour. Such a fast reaction in the UT cluster team exists because of mobile notifications which may lift administrators late at night, in order, to fix servers. Worth mentioning that we have much logging and have to separate flows for "access level logging" and "error level logging". It also contributes to reproducing errors on a local machine and fast debugging website related incidents, which is significantly important to users satisfaction.

As it was mentioned in previous chapters, the system backed up weekly, and save a state of all data and files can be quickly restored.

3.3.5 Testing

The project itself is small and does not have a lot of advanced features, yet. The primary focus was to make it user-friendly and easy to use, also reliable but it was fixed by moving the UT HPC cluster. The obvious thing to many developers nowadays to start writing tests first following test-driven development or/and behaviour-driven development approaches and then implement the functionality. The project had time and budget constraints (It had to be alive in less than three months of part-time work from requirements to a production). Therefore, the author has explored several options. Two articles which contributed heavily to the author's decision of testing technique are [Inca] and [Incb]. Both of articles discuss pros and cons of between two techniques. From my perspective the main pros which the author based his decision on are "most likely to find real user-issue" and "product does not have any advanced features which are hard to test" and tests were not meant to "run repeatedly too often" (no more than 1-2 times per month, when significant changes in functionality happened).

To summarize, the author thinks in a long term perspective automated tests are required for both front-end and back-end, and it should be one of the top priorities while the project is growing.

3.4 Conclusions

To sum up, the architecture chapter, the system seems to be composed of simple components, but behind the scene, it required much effort, many sources of information to make an appropriate decision, and expertise. The design and system overall can be and will be tuned over time. From experience gained during this project, the author can conclude that Flask framework was not an ideal choice for the back-end, the front-end can be separated with the back-end, and the work may be done in a more lean way. For example, the author assumes that the better option will be to have the whole system running from the beginning on the UT HPC cluster to make new deployments leaner and move from a big batch deliveries to small quantities.

4 Analysis of results

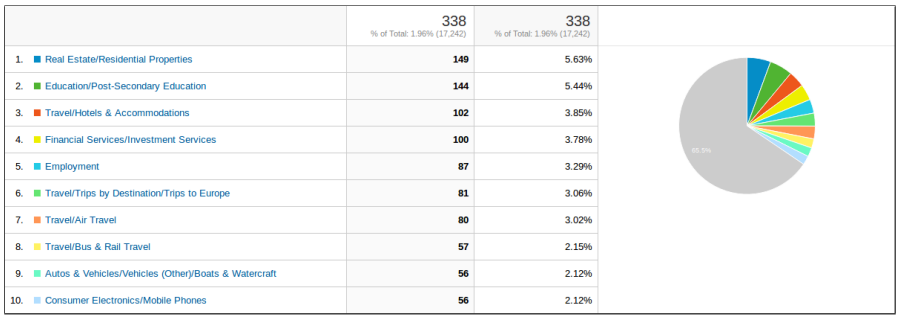
This section describes analytical tools which were used to evaluate the hypothesis, analyze data generated by users and shows performance and technical details regarding the product.

4.1 User demand analysis

To test the hypothesis whether people need such a tool for their everyday life, the author decided to use several tools:

- Google Analytics[17]
- Hotjar[18]
- surveys created in Google Forms[19]

Google Analytics is an exceptional tool to collect data about all aspects of the website usage. Still, it has a huge drawback as it does not have enough information on the user. We do not know who they are and how the use the website. Nevertheless, on figure 2 can be observed in-segment percentage of the active users.



	338 % of Total: 1.96% (17,242)	338 % of Total: 1.96% (17,242)
1. Real Estate/Residential Properties	149	5.63%
2. Education/Post-Secondary Education	144	5.44%
3. Travel/Hotels & Accommodations	102	3.85%
4. Financial Services/Investment Services	100	3.78%
5. Employment	87	3.29%
6. Travel/Trips by Destination/Trips to Europe	81	3.06%
7. Travel/Air Travel	80	3.02%
8. Travel/Bus & Rail Travel	57	2.15%
9. Autos & Vehicles/Vehicles (Other)/Boats & Watercraft	56	2.12%
10. Consumer Electronics/Mobile Phones	56	2.12%

Figure 2: The in-segment market of the platform users

Collecting period started on 16th April. The University of Tartu[20], Postimees[21],

[17] `https://www.google.com/analytics/?modal_active=none`. Last accessed at 2017-05-16
[18] `https://docs.hotjar.com/docs`. Last accessed at 2017-05-16
[19] `https://www.google.com/forms/about/`. Last accessed at 2017-05-16
[20] `https://www.ut.ee/et/uudised/keeletehnoloogid-valmistasid-uuendusliku-eestikeelse-masintolke-`
[21] The official website is `http://news.postimees.ee/`. The article about the project `http://www.postimees.ee/teema/neurotolge.ee`

and Delfi[22] wrote several articles about the project also it was shown on TV and noticed on the radio.

The statistics over a month period is drawn on figure 3.

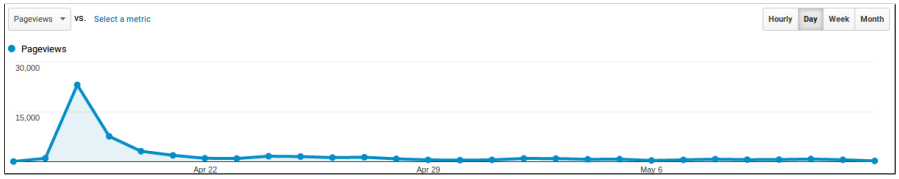


Figure 3: Page views over the month period

The statistic shows that during broadcasts and promotion campaign the website gained extreme popularity and up to now rating has plummeted from 20000 users per day to in average 400 per day. Another figure 4 shows statistics after 20th April inclusively (here and later in this chapter the author names this date as "the unbiased date"). The date was chosen based on the end of the active PR campaign and the extinction of the initial splash of visitors, in order to remove the bias factor.

From the period from the unbiased date to 15th May, the author can conclude that drop happened approximately with factor two and statistically on weekends users are much less active than during the weekdays. The first conclusion that comes to the mind is that the platform concept failed. To support or refute the "obvious" conclusion we need to look on figure 5. Therefore, even people use the website less, the average time per page grows. Naturally, this data is insufficient to make global conclusions, so let's look at even more statistics.

From one side, on figure 6 cohort analysis, the starting date comparing to user retention, we may assume that the project goes okay. From the other hand, on figure 7 on the tenth day so little people use the website, so it looks like an absolute failure of the project. From this two cohort analysis, themselves is not possible to make any sound and objective conclusions but they are indicators that the platform need to be analyzed in depth, and closer collaboration with end-users is required.

To understand the phenomenon which is observed in Google Analytics we need to collect feedback from users. It is time for Hotjar and user surveys unmask hidden problems. The author figured out that Google Analytics was insufficient in finding out users behavior and because of that on 12th May added two surveys

[22]The official website is `http://www.delfi.ee/`. The article about the website is `http://www.delfi.ee/teemalehed/neurotolge`

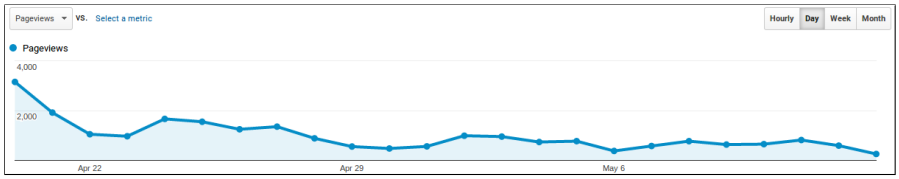


Figure 4: Page views starting from 20th April

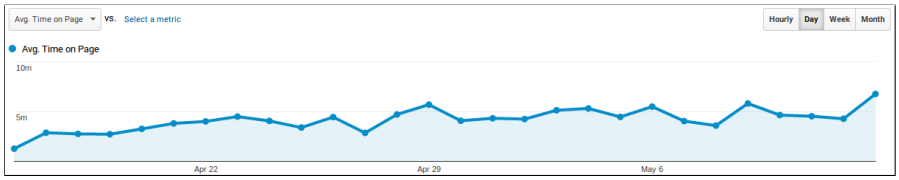


Figure 5: Average time per page for the month period

for Estonian and English speaking users. Observing behavior several days[23] and finding out that users do not want to fulfill surveys the author has added Hotjar tool which not only asks quick feedback but also records behavior, clicks and actions and allows to observe users; can be seen on figure 8. Also, Hotjar draws heatmaps which allow determining dead zones and UX flaws.

The author extracted heatmaps from Hotjar tool and on figure 9 is shown that desktop users use only main functionality "translate" and "play". Also, they usually use "play" mode if they are not satisfied with the UT translation. Tablet and mobile users behavior can be seen on figures 10 and 11. Mobile and tablet users mostly repeat desktop clients behavior.

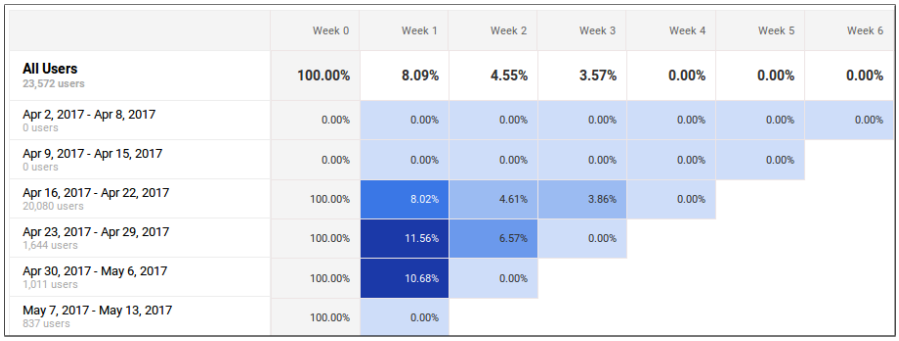
Based on the heatmaps the first UX flaw is too many clicks on "translate from" section. The initial conclusion which was made that Estonian to English pair has to be changed to English to Estonian as a default one, but observing users recordings of the page usage in IE and Microsoft Edge browsers turned up to have a bug which is also contributed to such heatmap's view. Since the project is the open-source platform, the author decided to use BrowserStack[24] because of their open-source projects support[25] to improve browsers compatibility.

To sum up, there is not enough data to make conclusions regarding users satisfaction, but based on the group of users the project has it is the tiny community.

[23]For two first days, only two users left their feedback. On the sixth day only seven users left their comments, and in fact, only five could be used.

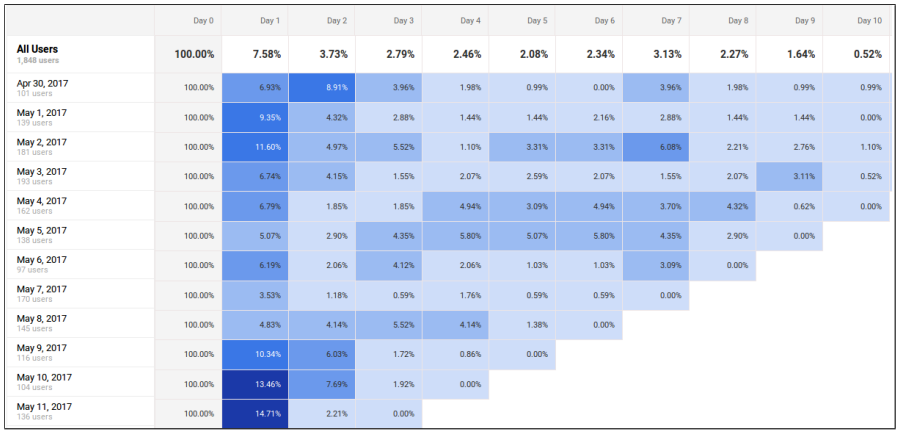
[24]The official website is `https://www.browserstack.com/features`. Last accessed at 2017-05-16

[25]`https://www.browserstack.com/contact?ref=open_source`



	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
All Users 23,572 users	**100.00%**	**8.09%**	**4.55%**	**3.57%**	**0.00%**	**0.00%**	**0.00%**
Apr 2, 2017 - Apr 8, 2017 0 users	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Apr 9, 2017 - Apr 15, 2017 0 users	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Apr 16, 2017 - Apr 22, 2017 20,080 users	100.00%	8.02%	4.61%	3.86%	0.00%		
Apr 23, 2017 - Apr 29, 2017 1,644 users	100.00%	11.56%	6.57%	0.00%			
Apr 30, 2017 - May 6, 2017 1,011 users	100.00%	10.68%	0.00%				
May 7, 2017 - May 13, 2017 837 users	100.00%	0.00%					

Figure 6: Cohort analysis date of usage start and user retention rate by weeks



	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
All Users 1,848 users	**100.00%**	**7.58%**	**3.73%**	**2.79%**	**2.46%**	**2.08%**	**2.34%**	**3.13%**	**2.27%**	**1.64%**	**0.52%**
Apr 30, 2017 101 users	100.00%	6.93%	8.91%	3.96%	1.98%	0.99%	0.00%	3.96%	1.98%	0.99%	0.99%
May 1, 2017 139 users	100.00%	9.35%	4.32%	2.88%	1.44%	1.44%	2.16%	2.88%	1.44%	1.44%	0.00%
May 2, 2017 181 users	100.00%	11.60%	4.97%	5.52%	1.10%	3.31%	3.31%	6.08%	2.21%	2.76%	1.10%
May 3, 2017 193 users	100.00%	6.74%	4.15%	1.55%	2.07%	2.59%	2.07%	1.55%	2.07%	3.11%	0.52%
May 4, 2017 162 users	100.00%	6.79%	1.85%	1.85%	4.94%	3.09%	4.94%	3.70%	4.32%	0.62%	0.00%
May 5, 2017 138 users	100.00%	5.07%	2.90%	4.35%	5.80%	5.07%	5.80%	4.35%	2.90%	0.00%	
May 6, 2017 97 users	100.00%	6.19%	2.06%	4.12%	2.06%	1.03%	1.03%	3.09%	0.00%		
May 7, 2017 170 users	100.00%	3.53%	1.18%	0.59%	1.76%	0.59%	0.59%	0.00%			
May 8, 2017 145 users	100.00%	4.83%	4.14%	5.52%	4.14%	1.38%	0.00%				
May 9, 2017 116 users	100.00%	10.34%	6.03%	1.72%	0.86%	0.00%					
May 10, 2017 104 users	100.00%	13.46%	7.69%	1.92%	0.00%						
May 11, 2017 136 users	100.00%	14.71%	2.21%	0.00%							

Figure 7: Cohort analysis date of usage start and user retention rate by days

4.2 User behavior analysis

This section concentrates exclusively on translations and statistics. One of the goals of building the platform was to collect crowd-source data.

On the state of 15th May 2017, 10537 votes were collected, in the voting 10051 sentences are unique. Approximately, each week 1500 votes are collected this number excludes biased first days of promotion company. Comparing to 15,4 million sentences corpora which currently are in use, this data can be utilized but logically will have less affection on the model, at the same time improved model should positively affect on users satisfaction. The gathered data may be applicable only at the end of the year if the website will be used with the same frequency. Counting from 15th May to the end of the year 32 weeks left, so no more than 48

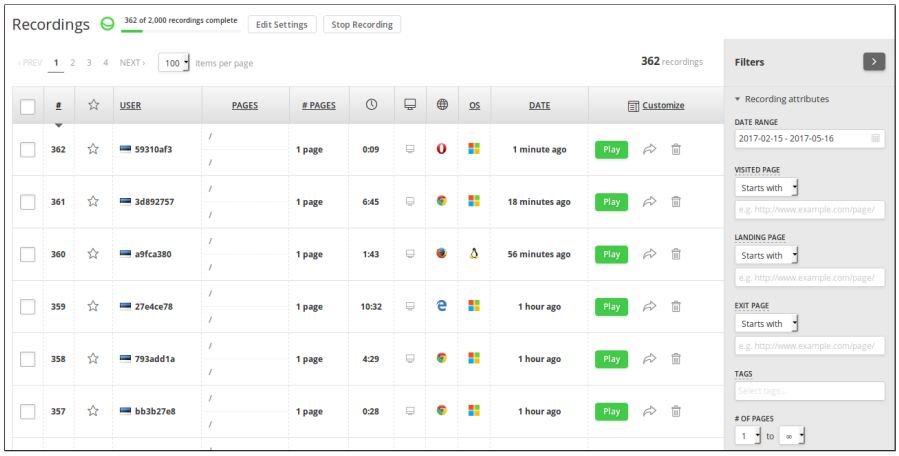


Figure 8: List of user behavior recordings

thousands of sentences will be collected[26], which is already a good result for the product which has many competitors on the market. It remains incomprehensible if idea with collecting data through crowd-sourcing works.

Below statistics which do not exactly show users behavior but rather result in this behavior.

From the launch of the website more than 150 thousands sentences were translated. Below is a list of the most translated sentences to Estonian in table 1 and from Estonian in table 2.

There is the percentage ratio of user votes per translator is represented on figure 12.

To sum up, to make crowd-source data collection possible we need to motivate users to vote more or add an option to add their translations if the reason why they do not vote is a quality of all translations.

4.3 MT performance analysis

Performance section shows the speed of the website, technical characteristics of browsers being used by clients. All data begins time countdown from 20th April to remove the biased factor of the first days.

There are several figures from Google Analytics. Average loading time for all browsers can be seen on figure 13 and for top used browsers on figure 14. There is no an easy answer for Firefox and Edge of being slower compared to Safari or Chrome. Nevertheless, there is an EdgeHTML issue tracker with reports about

[26] Assuming that data collection speed will continue to be constant in an average.

Frequency	Sentence
260	Meow...
72	Maru videos: men and machines in the mud
50	Why do people want to have sex first thing in the morning?
50	Twelve kenties of a sex story that makes me laugh
47	What do you say we move... to Turkey?
47	The joy of the wheel world vanished by one second
47	Project Bastard: part 12 - how to dispose of grey
47	North Korea threatened to sink the US aircraft carrier
47	Less insulting and more emotion gave Tartu the Bigbank love time
45	Video: the Mess was chosen to win the Barcelona Real, and he loses the home club

Table 1: Top 10 most frequent translations from English to Estonian

Frequency	Sentence
1298	tere
851	Tere
765	Pähe õpitud luuletust võib lugeda igaüks ja kellele tahes, kuid oma loodud värsid pühendatakse vaid ühele ainsale inimesele. Seetõttu on nad ülimalt väärtuslikud.
537	Parimat soovides
308	"Pähe õpitud luuletust võib lugeda igaüks ja kellele tahes, kuid oma loodud värsid pühendatakse vaid ühele ainsale inimesele. Seetõttu on nad ülimalt väärtuslikud."
279	"Pähe õpitud luuletust võib lugeda igaüks ja kellele tahes, kuid oma loodud värsid pühendatakse vaid ühele ainsale inimesele. Seetõttu on nad ülimalt väärtuslikud."?
260	Mjäu...
252	Parimat soovides,
200	Pähe õpitud luuletust võib lugeda igaüks ja kellele tahes, kuid oma loodud värsid pühendatakse vaid ühele ainsale inimesele. Seetõttu on nad ülimalt väärtuslikud
171	Parimat soovides[27]

Table 2: Top 10 most frequent translations from Estonian to English

Figure 9: Heatmap of user clicks for desktops

browser slowness[28] for Firefox and there are many similar questions on different forums[29]. Also, such a sluggishness may be because users do not update browsers frequently, so some versions contain bugs and these versions load longer affecting average loading speed. There is a reasonable conclusion that slow browsers have to be debugged, to prioritize them the author will be guided by figure 15 and the top priority will be Firefox.

Interesting statistics about devices can be seen on figure 16, 17 and 18. Based on this data Android and Apple mobile device users should have the highest possible service priority.

The distribution of users operating systems is presented on figure 19.

4.4 Conclusions

To summarize, the analysis shows that even in the early stages of the project there are no breakthrough results it still shows potential to collect significant amount

[28] `https://developer.microsoft.com/en-us/microsoft-edge/platform/issues/?page=1&q=slow`. Last accessed at 2017-05-16

[29] One of the examples `https://support.mozilla.org/en-US/questions/1011782`. Last accessed at 2017-05-16

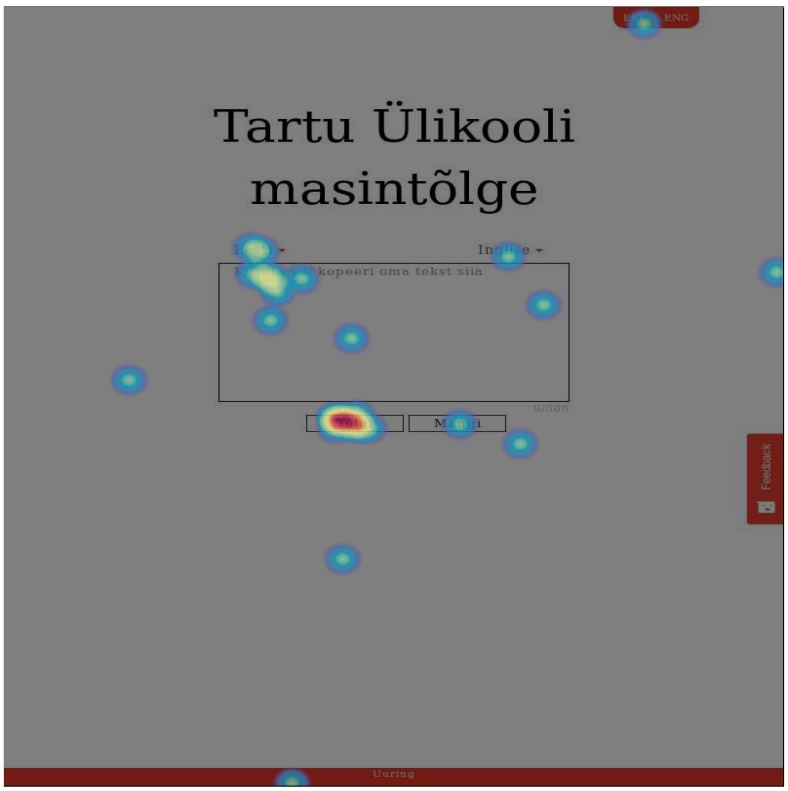


Figure 10: Heatmap of user clicks for tablets

of data before the end of 2017. The platform compatible amidst different devices, platforms, browsers which allows to have a share in the market. The figures 20 and 21 draws top channels. Social media and organic search are not fully developed and have many potentials. Also, based on the insight from early adopters many users expecting new language pairs which should attract an even bigger audience and contribute to the growth of the crowd-source database.

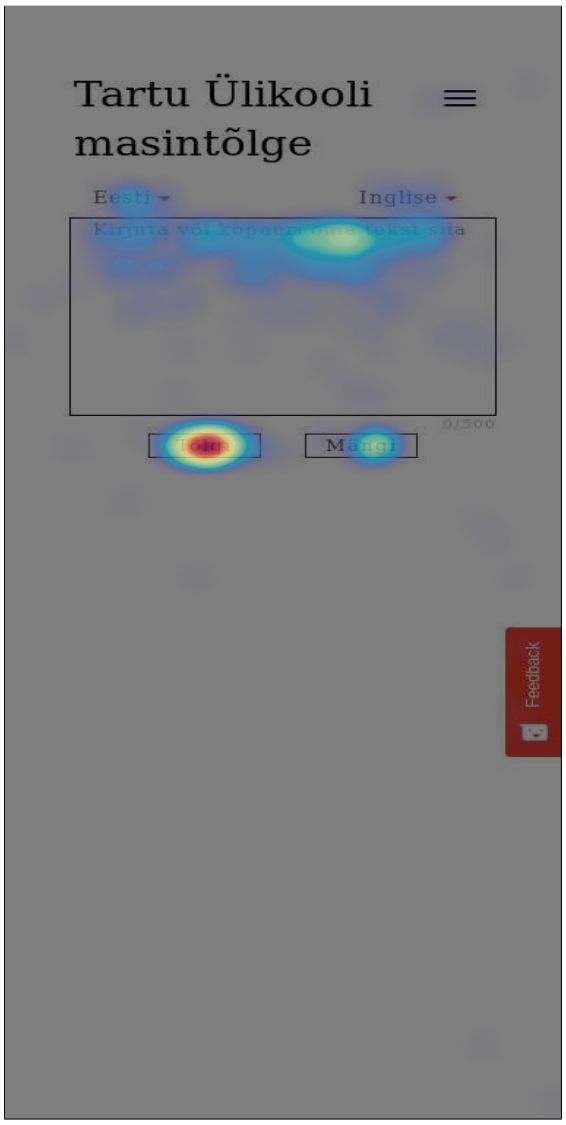


Figure 11: Heatmap of user clicks for mobiles

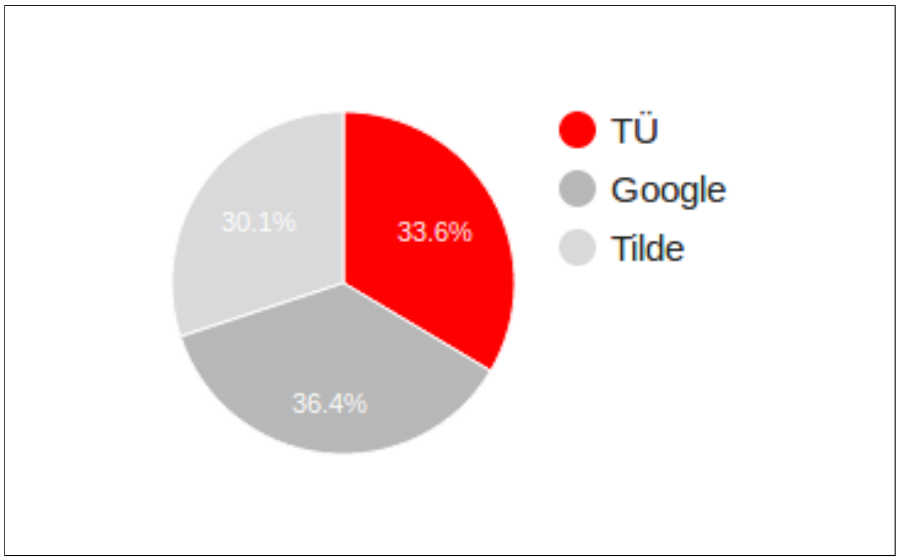


Figure 12: The percentage ratio of user votes per translator

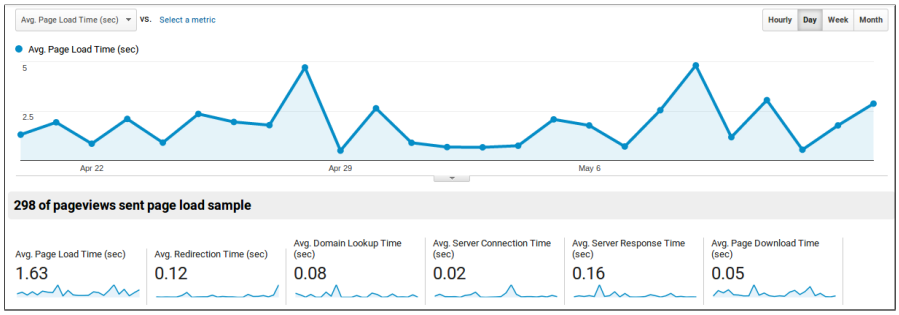


Figure 13: Average loading time of the website

Browser	Avg. Page Load Time (sec)	Pageviews	Bounce Rate	% Exit	Page Value
	1.63 Avg for View: 1.63 (0.00%)	23,815 % of Total: 100.00% (23,815)	81.01% Avg for View: 81.01% (0.00%)	72.40% Avg for View: 72.40% (0.00%)	$0.00 % of Total: 0.00% ($0.00)
1. Firefox	**3.91**	2,661 (11.17%)	83.15%	76.51%	$0.00 (0.00%)
2. Edge	**1.48**	545 (2.29%)	69.61%	51.93%	$0.00 (0.00%)
3. Chrome	**1.31**	14,350 (60.26%)	82.44%	75.54%	$0.00 (0.00%)
4. Internet Explorer	**0.97**	1,802 (7.57%)	54.65%	42.34%	$0.00 (0.00%)
5. Safari (in-app)	**0.86**	162 (0.68%)	87.86%	86.42%	$0.00 (0.00%)
6. Android Webview	**0.84**	473 (1.99%)	75.24%	64.90%	$0.00 (0.00%)
7. Safari	**0.60**	3,419 (14.36%)	82.45%	74.82%	$0.00 (0.00%)
8. Amazon Silk	**0.00**	2 (0.01%)	0.00%	50.00%	$0.00 (0.00%)
9. Android Browser	**0.00**	28 (0.12%)	96.30%	96.43%	$0.00 (0.00%)
10. Maxthon	**0.00**	13 (0.05%)	91.67%	92.31%	$0.00 (0.00%)

Figure 14: Average loading time of the website for most used browsers

	17,242 % of Total: 100.00% (17,242)	17,242 % of Total: 100.00% (17,242)
1. Chrome	**10,840**	62.87%
2. Safari	**2,558**	14.84%
3. Firefox	**2,036**	11.81%
4. Internet Explorer	**763**	4.43%
5. Android Webview	**307**	1.78%
6. Edge	**283**	1.64%
7. Opera	**261**	1.51%
8. Safari (in-app)	**140**	0.81%
9. Android Browser	**27**	0.16%
10. Maxthon	**12**	0.07%

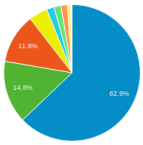


Figure 15: Most frequently used browsers

	17,242 % of Total: 100.00% (17,242)	17,242 % of Total: 100.00% (17,242)
1. desktop	**12,242**	71.00%
2. mobile	**4,382**	25.41%
3. tablet	**618**	3.58%

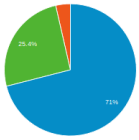


Figure 16: Devices are used by the website users

	5,000 % of Total: 29.00% (17,242)	5,000 % of Total: 29.00% (17,242)
1. Apple iPhone	**1,639**	32.78%
2. Apple iPad	**321**	6.42%
3. (not set)	**264**	5.28%
4. Samsung SM-G920F Galaxy S6	**120**	2.40%
5. Huawei ALE-L21 P8 Lite	**116**	2.32%
6. Huawei VNS-L21 P9 Lite	**100**	2.00%
7. Samsung SM-G935F Galaxy S7 Edge	**95**	1.90%
8. Samsung SM-G925F Galaxy S6 Edge	**77**	1.54%
9. Samsung SM-T555 Galaxy Tab A 9.7	**59**	1.18%
10. Samsung SM-G900F Galaxy S5	**55**	1.10%

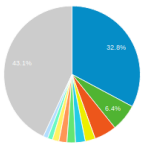


Figure 17: Most frequently used models of mobile devices

		5,000 % of Total: 29.00% (17,242)	5,000 % of Total: 29.00% (17,242)
1.	Apple	2,091	41.82%
2.	Samsung	1,252	25.04%
3.	Huawei	408	8.16%
4.	Sony	368	7.36%
5.	(not set)	264	5.28%
6.	LG	120	2.40%
7.	Google	74	1.48%
8.	Asus	65	1.30%
9.	OnePlus	63	1.26%
10.	Lenovo	58	1.16%

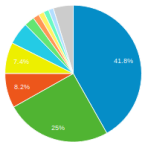


Figure 18: Most commonly used brands of mobile devices

		17,242 % of Total: 100.00% (17,242)	17,242 % of Total: 100.00% (17,242)
1.	Windows	10,677	61.92%
2.	Android	2,848	16.52%
3.	iOS	2,091	12.13%
4.	Macintosh	1,352	7.84%
5.	Linux	261	1.51%
6.	Windows Phone	9	0.05%
7.	Chrome OS	4	0.02%

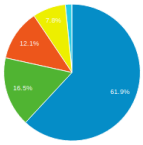


Figure 19: Distribution of operating systems usage by users

	Acquisition			Behavior		
	Sessions ↓	% New Sessions	New Users	Bounce Rate	Pages / Session	Avg. Session Duration
	17,242	31.09%	5,360	81.01%	1.38	00:01:34
1 Direct	7,638			81.67%		
2 Referral	5,759			80.71%		
3 Organic Search	2,896			80.32%		
4 Social	949			79.56%		

Figure 20: Top channels of customer attracting

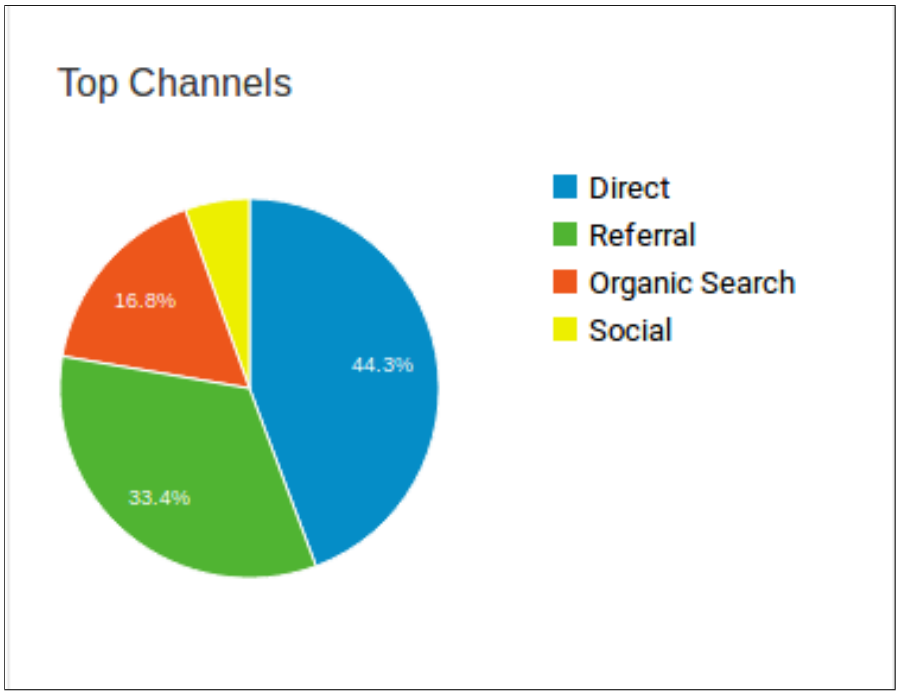


Figure 21: Top channels of customer attracting as the pie chart

5 Conclusions

In this chapter, the author wants to summarize different aspects of the thesis completed and revisit problems and hypotheses set in section 1.1.

Problem. There is no open-source platform for machine translation system with availiability to collect data from end-users.

The author has created such a platform. Project is open-source and improves continually based on the end-user feedback. Tables 1 and 2 show examples of collected sentences.

Hypothesis 1. Do users need such a system for everyday usage?

Based on Google Analytics and another analytical tool users do need the system. In the same time, only quite a few use it on an everyday basis. Therefore, this hypothesis remains unrequited.

Hypothesis 2. How fast can evaluation be collected?

This hypothesis needs more precise formulation but assuming that 100 thousand sentences are the goal, they may be gathered in a less than two years, based on estimation provided in section 4.

6 Future work

This chapter opens the veil of the project backlog. Ideas contain all contemporary development techniques and methods which were not included in the thesis because of time constraints or lack of physical resources.

6.1 Project "dockerization"

There are several points why project "dockerization" is important. Firstly, it helps development teams which are using different operating systems deploy environments both locally and to continuous integration platforms such as Codeship, Jenkins. Secondly, a docker image is helpful for scaling applications. It will be crucial for the project when it will reach a point when it had to be scaled rapidly. Finally, if at some point the website will move to another server, deployment will be technically simpler.

The article [Wil] draws all concerns because of which this task was put to the backlog. Uncertainty about the security of API keys safety, tricky and custom editions in the deployment of a docker image. Nevertheless, the author thinks "dockerization" should be implemented with a Docker or analogs software container (e.g. Rocket). There is an answer on experts-exchange for [Bil] which supports the author's opinion and explains all advantages of Docker usage.

6.2 Automated testing

With the increasing size of the project, the difficulty of testing a new feature grows. To simplify this process and make integration with continuous deployment solutions easier tests have to be automated. The plan is to start from unit-testing and end-to-end feature tests.

References

[Agu] Agus. http://www.hongkiat.com/blog/bootstrap-alternatives/. Last accessed at 2017-05-07.

[Ano] Anonymous. https://www.quora.com/How-do-Rails-vs-Flask-vs-Pyramid-compare-as-web-frameworks. Last accessed at: 2017-05-05.

[BBCB+13] Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv*, 2014.

[BCF+16] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Bil] James Bilous. https://www.experts-exchange.com/questions/28619388/In-what-scenarios-is-Dockerization-used.html. Last accessed at 2017-05-10.

[BPPM93] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993.

[Bus] David Bushell. https://www.smashingmagazine.com/2011/06/understanding-copyright-and-licenses/. Last accessed at 2017-05-16.

[Car] Brad Carleton. https://lostechies.com/bradcarleton/2014/03/25/frontend-backend-gotta-keepem-separated/. Last accessed at 2017-05-09.

[com] Slantś community. https://www.slant.co/versus/1233/1398/~ruby-on-rails_vs_flask. Last accessed at: 2017-05-05.

[Dev] Alexis Deveria. http://caniuse.com/#feat=websockets. Last accessed at 2017-05-12.

[Dwy] Gareth Dwyer. https://www.codementor.io/garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v. Last accessed at: 2017-05-05.

[FGRN+11] Mikel L. Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. Apertium: A free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144, June 2011.

[Fou] Python Software Foundation. https://docs.python.org/2/library/queue.html. Last accessed at 2017-05-08.

[GLI14] Shinsuke Goto, Donghui Lin, and Toru Ishida. Crowdsourcing for evaluating machine translation quality. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA).

[Har] Chris Harrington. https://www.codementor.io/reactjs/tutorial/reactjs-vs-angular-js-performance-comparison-knockout. Last accessed at: 2017-05-07.

[Hus] Maarten Hus. http://dontpanic.42.nl/2014/10/the-case-for-separating-front-and-back.html. Last accessed at 2017-05-11.

[Inca] Apica Inc. https://www.apicasystem.com/blog/automated-testing-vs-manual-testing/. Last accessed at 2017-05-08.

[Incb] Base36 Inc. http://www.base36.com/2013/03/automated-vs-manual-testing-the-pros-and-cons-of-each/. Last accessed at 2017-05-08.

[Inn] Innolution. http://www.innolution.com/resources/glossary/component-team. Last accessed at 2017-05-11.

[Inta] Ecma International. https://www.ecma-international.org/ecma-262/5.1/. Last accessed at 2017-05-08.

[Intb] Ecma International. https://www.ecma-international.org/ecma-262/6.0/. Last accessed at 2017-05-08.

[Joh] Simone Johnson. https://quickleft.com/blog/six-reasons-we-split-front-end-and-back-end-code-into-two-git-repositor Last accessed at 2017-05-09.

[KOM03] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[Kon] Aaron Koning. https://blog.safe.com/2014/08/websockets-ajax-webhooks-comparison/. Last accessed at 2017-05-12.

[Kor] Eugeniya Korotya. https://hackernoon.com/reactjs-vs-angular-comparison-which-is-better-805c0b8091b1. Last accessed at 2017-05-07.

[Ron] Armin Ronacher. http://jinja.pocoo.org/docs/2.9/faq/#how-fast-is-it. Last accessed at 2017-05-11.

[Teca] Segue Technologies. http://www.seguetech.com/ajax-technology/. Last accessed at 2017-05-12.

[TECb] TECHSTUFF. http://computer.howstuffworks.com/question435.htm. Last accessed at 2017-05-16.

[Use] User-182633. https://www.quora.com/Why-does-it-make-sense-to-separate-front-end-from-back-end. Last accessed at 2017-05-09.

[Wil] Jason Wilder. http://jasonwilder.com/blog/2014/10/13/a-simple-way-to-dockerize-applications/. Last accessed at 2017-05-10.

[Zak] Nicholas Zakas. https://www.smashingmagazine.com/2013/01/starting-an-open-source-project/. Last accessed at 2017-05-16.

Non-exclusive licence to reproduce thesis and make thesis public

I, Dmytro Chasovskyi (date of birth: 15th of June 2017),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Platform for a neural machine translation system demo and user data collection

supervised by Mark Fishel

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu/Tallinn, 18.06.2017