browser-based multilingual translation

bergamot

Horizon 2020 Research and Innovation Action
Grant Agreement No. 825303

https://browser.mt

Deliverable D6.1: Basic Firefox Integration

Lead author(s): Kelly Davis (Mozilla)
Contributing author(s): NA
Internal Reviewer(s): Roman Grundkiewicz (UEDIN)

Work Package: WP6
Type of Deliverable: Other
Due Date: 30 September 2019
Date of Submission: 30.09.2019
Current Version: 1.0

Document History

Version	Date	Changes
1.0	30.09.2019	Original Submission

Executive Summary

The Bergamot project will deploy machine translation client-side on desktop computers as an extension to Firefox. As an initial step towards this goal, Deliverable 6.1 integrates server-side machine translation into Firefox, translating web sites on the fly by sending the web site's text to a remote translation server. The primary components of this deliverable are contributions to the code base of the open-source Firefox browser (available at https://github.com/browsermt/firefox/tree/D6.1-Basic-Firefox-Integration), establishment of a server end-point exposing the open-source Bergamot machine translation system (available at https://github.com/marian-nmt/marian-dev), and the integration of these two components. A video of the system translating a web page is available at (https://youtu.be/ptmLzVeU0dk?vq=hd2160).

Contents

1 Introduction

The Bergamot project will add client-side machine translation to the Firefox browser. Unlike current cloud-based options, running directly on users' machines empowers citizens to preserve their privacy and increases the uptake of language technologies in Europe in sectors that require confidentiality. However, integrating the complexities of machine translation into Firefox client-side is a multi-step process. This deliverable is the first step in this process.

In this deliverable we integrate the *server-based* Bergamot machine translation engine with the Firefox browser. As a result of this integration, when a user browses to a page *not* in their default language, the top of their browser window displays a Translation Infobar that gives the option to translate the web page from the source language to the user's default language, see Figure 1.

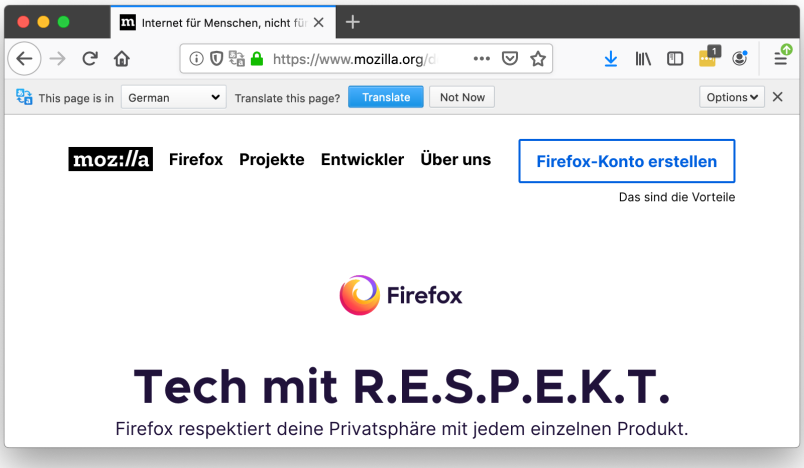

Figure 1: Screenshot of Firefox displaying the option to translate.

The user can then select to translate the web page if so desired. If they indicate they want the web page translated, the web page is decomposed into its various textural components and each is sent to the Bergamot machine translation server for translation. Once these are translated by the server, these textural components are reassembled to create a translated version of the web page with the same layout as the original, see Figure 2.

2 A Brief History of Firefox Translation

Firefox translation has had a long and fruitless history within Mozilla. There have been at least two different efforts over the last 5 years to integrate translation into the browser. Both have failed.

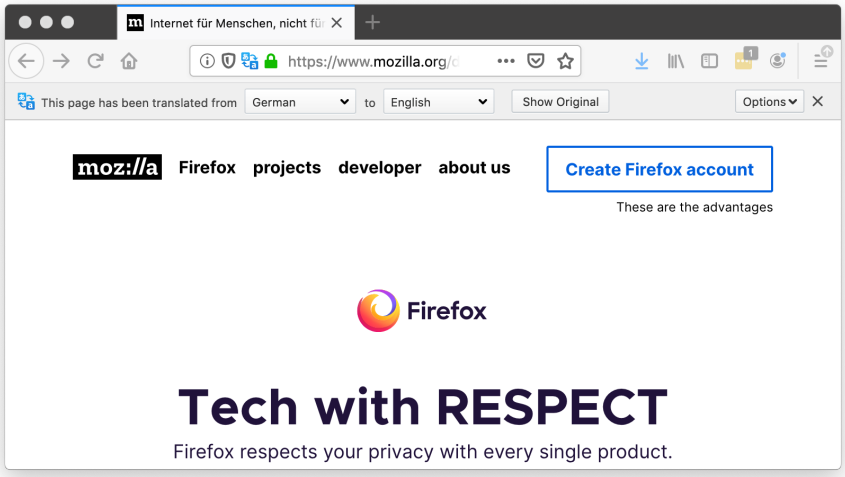

Figure 2: Screenshot of Firefox displaying a translated web page.

The Intellego Project, the first effort which started at the beginning of 2014, planned to "provide a single platform for (machine translation) engine developers and a unified web service that hosts a number of different language pairs/engines/implementations in the back end," allowing users to "select from a number of open MT engines based on the most prominent MT methodologies in order to find the best target MT output for their on-the-fly translation."

The approach of the Intellego Project, while laudable, had many flaws, several of them obvious several of them more subtle. For example, the Intellego Project took no consideration of how the costs of the servers would be handled. Shouldering the costs of a small number of translations is not problematic. However, in scaling this to the entire user base of Firefox, server costs quickly become a major issue. The Intellego Project also did not take maintenance into consideration. Maintaining a single internal translation engine for all the languages Firefox supports is a huge undertaking. Maintaining numerous external translation engines for all the languages Firefox supports is basically an impossible task when one wants maintain quality and operate at scale. Another problem, the Intellego Project didn't fully consider the motivations of external parties with translation engines. A large percentage of these external parties with high quality engines would understandably want to monetize their engines, reaping the benefits of their investments in time and money. The Intellego Project assumed that these parties would, without any monetary compensation, provide access to their engines and models. One can explore further issues with the Intellego Project, but these give a flavor of the reasons the project failed.

The second effort, which started in the middle of 2018, planned to integrate Google translation services into Firefox as part of a larger business deal switching Mozilla's infrastructure over to Google's Cloud Platform (GCP). This effort was more grounded in the realities of what is required to sup-

port a translation engine at scale. However, the specifics of the larger business deal shifted and, despite Mozilla's move towards GCP, use of Google's translation service was not included in the deal. Firefox was left once again without any translation service.

So currently Firefox does not support automatic translation of web pages either through a "unified web service that hosts a number of different language pairs/engines/implementations in the back end" or through Google's translation services. In this regard Firefox is behind Google's Chrome.

3 Integration of Translation into Firefox

An heirloom which is gifted to Bergamot by this history of Firefox translation failures is a Firefox code base that has many of the primitives required to integrate server based translation. This existing code base serves to make basic Firefox integration with the Bergamot translation server much simpler.

The Bergamot translation server used for the initial integration is a German-English model, a part of the Edinburgh's submission to the News Translation Task at the Workshop on Machine Translation 2019 (Bawden et al., 2019). The model has been developed with the focus on translation quality using the Marian NMT toolkit (Junczys-Dowmunt et al., 2018a).

The code with the initial Firefox integration is available from: https://github.com/browsermt/firefox/tree/D6.1-Basic-Firefox-Integration

3.1 Basics

The primitives already within Firefox reduce the task of introducing a new Bergamot server based translation engine to the following set of four steps:

1. Create new JavaScript class that translates using the Bergamot server. In our case the class is `BergamotTranslator` defined in `BergamotTranslator.jsm`.
2. Register `BergamotTranslator.jsm` with the browser in `browser_all_files_referenced.js` and `moz.build`.
3. Set the value of the pref `browser.translation.engine` to the new translation engine, in this case `Bergamot`.
4. Set the default translation engine to `Bergamot` in `Translation.jsm`.

All of these steps are relatively trivial except the first step which is where the bulk of this deliverable focused.

Creating a new JavaScript class that translates using the Bergamot server involves creating a class that executes the following three steps:

1. Splitting the web page into textural elements to be translated.
2. Batching these textural elements into parallel requests, for performance sake, that are sent to the Bergamot translation server.
3. Asynchronously receiving results from the Bergamot server and placing the translated textural elements into the correct positions in the web page.

Implementation of a JavaScript class that executes the previous three steps was not a difficult task. However, the implementation exposed several interesting edge cases that we hadn't anticipated.

3.2 Advanced

In particular, the implementation is tasked with translating textural elements of the form

```
<p>Welcome to <b>Mozilla's</b> website</p>
```

to say Portuguese, which results in

```
<p>Bem-vindo à página da <b>Mozilla</b></p>.
```

The Bergamot translation server is currently only capable of translating pure text, "a" to "z" and "A" to "Z" along with punctuation. It is not capable of handling HTML tags. So the Firefox browser has to send pure text

```
Welcome to Mozilla's website
```

to the Bergamot translation server and receive pure text

```
Bem-vindo à página da Mozilla
```

in return and then re-insert the HTML tags. However, a moment's thought will show that this is impossible for a browser to do when it has no linguistic knowledge, for instance in the form of word alignment between input and translated sentences.

In translating "Welcome to Mozilla's website" to "Bem-vindo à página da Mozilla" the word order changed, "Mozilla" moved to the end, and "Mozilla" is no longer possessive. So, for the browser to re-insert the `<b>` tag around "Mozilla" it would have to know that word order changes in this sentence and "Mozilla" is no longer possessive. But, as the browser has no linguistic knowledge, this is an impossible task.

A possible solution, which we did not implement in this basic integration, is to train the Bergamot translation server to be able to handle HTML tags. So, for example, the browser would then send

```
<p>Welcome to <b>Mozilla's</b> website</p>
```

and receive

```
<p>Bem-vindo à página da <b>Mozilla</b></p>
```

in return.

Another solution is stripping and re-inserting HTML tags into the translated text based on word alignment. The word alignment can be generated with a word aligner, e.g. FastAlign (Dyer et al., 2013), or on-the-fly during translation, with the latter being preferred as this does not add an extra computational cost.

4 Learnings

The most important learnings gleaned from this basic Firefox integration were concerned with sentence tokenization, how we handle HTML tags, and with the general speed of translation.

4.1 Sentence Tokenization

The Firefox browser does not have any knowledge of a language's syntax. So, in particular, it does not know how to tokenize text into sentences. Thus, when presented with text of the form

```
It was a pleasure to burn.  It was a special pleasure to
see things eaten, to see things blackened and changed.
```

it does not know how to tokenize this into two sentences. So when sending this text to the server it sends the sentences together as one uninterpreted sequence of characters.

The NMT engine, however, performs best when it is translating once sentence at a time, not a sequence of sentences. So to optimize translation quality we had to find a means of delivering single sentences to the NMT engine.

The solution we came up with was introducing a server-side sentence tokenizer that tokenized the uninterpreted sequence of characters passed from the client into sentences. The solution keeps the client, as it should be, ignorant of a language's syntax and concentrates any knowledge of a language's syntax, as it should be, with the NMT engine.

4.2 HTML Tags

The Firefox browser does not have any knowledge of a web page's semantics. It is concerned only with syntax. Furthermore, the Firefox browser should not have any knowledge of a web page's semantics. So, when presented with text of the form

```
Do not touch
```

it views this text as an uninterpreted sequence of characters, with no semantic meaning. Thus, when such a text is translated to say French

```
Ne touche pas
```

it has no knowledge of the correspondence between the uninterpreted sequence of characters "Do not touch" and the uninterpreted sequence of characters "Ne touche pas".

If HTML tags are introduced

```
<b>Do not</b> touch
```

the resulting translation should be of the form

```
<b>Ne</b> touche <b>pas</b>.
```

However, as the Bergamot translation server does not handle HTML tags, the Firefox browser is forced to pass "Do not touch" to the server, receive "Ne touche pas" as a result, and re-insert the various HTML tags. But, without semantic knowledge of the text, this is impossible. Thus, there is a fundamental problem that must be addressed in regards to HTML tags.

To learn a bit about how this problem is handled by other production translation engines, we tested various engines with this example

```
<b>Do not</b> touch
```

to see how they handled HTML tags. Currently Google does the following

```
<b> Ne pas </ b> toucher
```

Bing this

```
<b>Ne touchez pas </b>
```

Biadu this

```
< b > ne pas toucher
```

DeepL this

```
<b>Do not</b> toucher.
```

Basically all production engines are imperfect in some way, with Google seemingly the least problematic.

4.3 Translation Speed

Another issue which was exposed as a result of this initial Firefox integration was that translation speed will be an issue. To some extent this is a problem of our basic setup, a single server using a single GPU to do translation, but it is also clear that this will be a real issue we have to address.

In this basic integration we attempted to speed translation by splitting web pages into textural elements, batching these, and sending these batches to the Bergamot translation server in parallel. This allowed for batches to be sent to the GPU keeping the GPU highly utilized. This helped some, but the translation speed still left something to be desired.

One obvious solution to this is to have N GPU's translating for a single Bergamot translation server endpoint. Thus would increase throughput by a factor of N. However, it would also increase costs by that same factor, which is not ideal when thinking about scaling this to a large user base.

An important thing to note is that the current neural machine translation system behind the Bergamot translation server that is used for this initial integration has been optimized only for translation quality, not for decoding speed. In the future, we will use student models developed with the teacher-student method (Junczys-Dowmunt et al., 2018b) and other optimizations (Work Package 5), which allows for significant speed improvement, up to 20 times on a single GPU.

Another possible solution, and the one Chrome uses, is to translate web sites in a "lazy manner". By this we mean, send all the strings to be translated to the server but prioritize those strings that are user visible so they are completed first. Doing this it appears to the user as if all of the text is translated almost immediately. However, what the user does not know is that all the strings they are not currently viewing are not yet translated.

5 Conclusions

The basic integration was a success. We have an end-to-end system up and running with a good "first cut" at all of the key components involved in Firefox integration. In addition, we exposed some problems which we will

have to address for the final integration and exposed them early enough in the process that we can actually address them for later integrations.

References

Bawden, Rachel, Nikolay Bogoychev, Ulrich Germann, Roman Grundkiewicz, Faheem Kirefu, Antonio Valerio Miceli Barone, and Alexandra Birch. 2019. "The university of edinburgh's submissions to the wmt19 news translation task." *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, 103–115. Florence, Italy.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith. 2013. "A simple, fast, and effective reparameterization of IBM model 2." *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 644–648. Atlanta, Georgia.

Junczys-Dowmunt, Marcin, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018a. "Marian: Fast neural machine translation in C++." *Proceedings of ACL 2018, System Demonstrations*, 116–121. Melbourne, Australia.

Junczys-Dowmunt, Marcin, Kenneth Heafield, Hieu Hoang, Roman Grundkiewicz, and Anthony Aue. 2018b. "Marian: Cost-effective high-quality neural machine translation in C++." *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, 129–135. Melbourne, Australia.