Final Project Proposal:

Developer-Friendly Hooks to Moses Server; User-friendly Interface to Translation Engine

Jordan Matelsky — jmatels1
March 25, 2016

Introduction

Moses currently functions as a standalone process, and while it runs on a server — capable of accepting requests via HTTP request — it does not prioritize serving a public-facing API (namely, a JSON interface for RESTful calls). This project aims to rectify this deficit, not only adding a JSON endpoint framework to Moses, but also implementing standalone projects that utilize this framework to bring machine translation (via Moses) to (1) website translation on-page; (2) realtime voice 'chat' across language barriers.

Prior Work

Moses supports on-demand web-translation using a page-download functionality, available at this link. However, this functionality is largely reserved for those familiar with the Moses software: In the words of this project page, this system's target audience looks for those requesting "a proof-of-concept type of demonstration".

This project aims to incorporate this style of translation into a realtime Chrome extension that functions similarly to the existing Google Translate extension script available here.

In addition, this project aims to illustrate alternative uses of this same API interface by supporting the development of a Moses-based realtime speech translation platform, a la Skype's translation system.

Proposal

This project aims to provide a public-facing API interface for a Moses server to improve accessibility of this platform to non-programmers or non-developers. As a proof-of-concept, I intend to develop web-friendly (i.e. user-friendly) apps that utilize this API, enabling such services as realtime speech translation (via speech-recognition API), according to the proposed project on the Moses suggested-projects "Get Involved" page here. This system will use a standalone Moses server as described in the documentation to serve as a backend, while simple, low-overhead webpages will serve as the user-facing portal.

This serves to bring Moses translation to the forefront of usability in translation engines, rather than existing solely in researcher- or developer-usable spaces.

Technical Challenges

This project will focus on making the public-facing API more robust, while also developing a front-end (JavaScript) library to improve ease of development with the Moses machine translation engine. Current API specs already exist — a la the Google Translate API — which can be extended to promote Moses-specific functionality. This project will focus on improving asynchronous API calls for the Moses service, without sacrificing efficacy.

I predict that this project will involve both C++ (Moses-native) and Node (JS) development, and will span both web-development and machine-translation-specific design challenges.