

PiTranslator

Spencer Gautreaux

Matan Broner

9 December 2021

Executive Summary

Real time language translation is an important area of research. As the world becomes more interconnected, there is a growing interaction between the disparate groups. This project seeks to address these issues by removing the language barrier between these groups. We outline and build a system for near-real-time translation that can run on low power hardware. Utilizing the widespread Raspberry Pi platform and high-availability Google Cloud, users can leverage the latest advances in AI to get quality real time translation. This will remove language barriers that the users may experience without the need to expensive hardware.

Introduction

Background

TODO

Needs Statement

TODO

Goals and Objectives

TODO

Design Constraints and Feasibility

TODO

Literature Review

Stepes

Stepes is a commercially available solution for “fast and professional IoT Translation services”. Their goal is to allow IoT products (and companies thereof) to reach new markets through the use of translation services. As a commercial product, there is not a ton of insight in the implementation, however some details are still available. For example, Stepes’ website describes their service as “Ai-enabled, cloud powered”. This is similar to the product we develop in this project. However, Stepes also employs human translators. The role of these translators is unclear. By contrast, our system utilizes no human oversight on the translation process, a feature which helps improve timeliness of translation and security of the translation process. Stepes also requires a commercial contract (request for quote) whereas our system is built on open source api and hardware and runs on the widely available google API. This increases the transparency of our system and allows users to modify the system to satisfy their own requirements.

An IoT Technology for Development of Smart English Language Translation and Grammar Learning Applications

This paper, by Jiang et. al. outlines processes and problems associated with IoT translations. They explore this problem in the context of English instruction and teaching. Furthermore, the aggregate the data and discuss the potential to apply this data to develop and refine the curriculum around learning english as a second language. The first part, the challenges of IoT helps to refine the needs statement for this project. For example, the need to rapidly translate within a classroom environment is a subset of our larger need

statement. Likewise, in the classroom environment, cost is a major factor; it is untenable for a classroom to deploy expensive, high powered, compute devices to each student. The latter part of the paper, focused on english curriculum development, is less immediately relevant. However, it poses an interesting future work for this project. With enough scale, the ideas discussed therein could be realized; if enough users are working with the PiTranslator, a significant portion of aggregate data could be collected. This data could be applied to develop and refine the language learning experience in the manner outlined by Jiang et. al. It is an interesting possibility, but not one we explored in this project.

Google Translate API

We intend to build our product on top of the Google Translate API. As such, it is necessary to discuss its strengths and weaknesses. Principally, the Google Translate API (hereafter GT API), is itself a commercial product in the Google Cloud product offering. For this reason, it is highly available, with near constant uptime. This means that the service, for the most part, can be taken as available. Unfortunately since this is a near turn-key solution, the GT API represents a single point of failure system; should it go offline our product ceases to function. This is unlikely, but a possibility nevertheless.

The GT API leverages the massive compute power of Google Cloud to apply machine learning and produce consistently high-quality and accurate translations. Being in the Google Cloud, GT API is continually and transparently updated to ensure that results are of the highest quality every time.

Proposed Work

Evaluation of Alternative Solutions

1. **Local Compute:** One major downside of our solution is the reliance upon the Google Cloud. This represents a single point of failure, which is not ideal in an IoT system. It then must be asked if the translation work can be moved to the device. This introduces a couple new problems. Principally, this means the translation device must have more powerful compute, increasing device costs. This also increases the power requirement, decreasing runtime, if the device is deployed in a battery-operated manner. Finally in doing so, the devices may begin to move out of version with each other. Without a centralized authority providing the translation, the results of a translation will depend on the version(s) used by the devices involved in the translation. The devices would need to be able to interface across versions or update their versions at the edge, increasing complexity of the overall system.
2. **Build our Own Translation Engine:** Much like the local compute issue, to build and train a custom translation engine would require an untenable amount of processing. This problem is compounded if the data collection is done at the edge and all translation devices are required to synchronize their datasets. The scale of storage and compute required goes against the IoT device fundamentals. For that reason, relying on a pre-collected and trained AI model from a central authority (in this case Google Cloud), greatly simplifies the compute requirements of the devices deployed at the edge.
3. **Develop a “Closed Form” Translation Model:** Conceptually it is possible to develop an algorithm which performs translations without the need for Artificial Intelligence. If possible, this may significantly reduce the data requirements and this allow for the simplification of the edge compute. However, there is no such program that exists for doing so; this is very much an active area of research. Furthermore, if such a program did exist, it would need ongoing updates. Language is an evolving entity and thus the translation units would need to continually update their translation Algorithm. This reintroduces problems with differing translation versions as discussed previously.
4. **Make the Edge Compute Simpler:** A RaspberryPi is by no means the cheapest micro-controller available on the market. However, it is cheap enough and provides a rich feature set that helps the product. It has a large amount of support and natively integrates WiFi, Bluetooth, and USB, which allows it to interface with a wide variety of peripherals. Changing to a simpler device, while perhaps saving some money, complicates these issues.
5. **Employ a Human Translator:** While a human translator would provide excellent translation quality, higher than that of any computer program, they are expensive in terms of financial and (caloric) energy resources. They have significant downtime wherein they need to recover mentally and are in an

extremely finite supply. For these reasons, it is untenable to scale human translators.

Design Specification

TODO

Approach for Design Validation

TODO

Engineering Standards

Project Management

TODO

Weekly Schedule of Tasks, Pert and Gantt charts

TODO

Economic Analysis

TODO

Itemized Budget

Item	Cost	Qty	Description
Raspberry Pi 3B+	34.99	2	Compute Module for Raspberry Pi
SD Card	7.99	2	Store OS for Raspberry Pi
Headset Microphone Combo	26.99	2	Audio Input and Output
Battery	19.99	2	Provide Power for the Devices

Alternatively, a Raspberry Pi Zero W could be used. The per unit cost is then reduced from 34.99 to 9.99

References

Stepes. <https://www.stepes.com/iot-translation-services/> (accessed Dec 5. 2021)

Jiang, Y., Sabitha, R. & Shankar, A. An IoT Technology for Development of Smart English Language Translation and Grammar Learning Applications. Arab J Sci Eng (2021). <https://doi.org/10.1007/s13369-021-05876-1>

TODO

Appendices

Product Datasheets

TODO

Bio-Sketch

Spencer Gautreaux

TODO

Matan Broner

TODO