Translating English to Math

Dylan Orris

October 23, 2018

Contents

1 Introduction

Over the past decade, natural language processors have become more and more commonplace. In late 2011, Apple introduced their digital assistant, Siri, to iOS. Siri was the successor to voice control, an iOS feature that allowed the user to interact with their phone using their voice, though in both an extremely limited fashion and rather unreliably. Siri was powered by a much better voice recognition model. Not only could it process many commands into actions, but it could even ask for clarification based on context. Since then, Google introduced the Google Assistant to its Android and ChromeOS platforms, Amazon released smart home products powered by Alexa, and Siri has been ported to MacOS and WatchOS. All of these digital assistants rely on understanding the commands of users, and the only way this is possible is through natural language processing.

From Q2 2017 to Q2 2018, smart speakers, the quintessential aspect of the smart home to many, had shipments grow 187%. [**canalys**] This growth is expected to continue in the coming years, with smart speakers and other smart home technologies becoming an enormous industry. As this industry expands, there will be greater and greater access to these systems among the general population. Smart devices offer more than simply convenience, they could become an amazing source of educational opportunities which are not otherwise available for many. For example, what if it was possible for a student to ask their smart screen why a mathematical claim was true, and that system could respond by displaying the proof?

While we do not use smart devices in this project, we create a front end interface for students to more easily investigate mathematical questions, without the barrier of technical language.

1.1 Problem Statement

Automated theorem proving systems are a powerful tool which can allow a student to check their work or a researcher to prove a claim which could be extremely time consuming if done by hand. Unfortunately, these systems are not trivial to use, with the use of any such system requiring the user to learn the proper method for input to that system. As a result, these systems are quite intimidating to learn, and once one learns one such system, it is easy to be stuck using what is already known, even if better options may be available. A wonderful solution to this problem would be leveraging computational power to convert a human language request, e.g. "Prove that the length of the third side of a triangle is determined by the two other sides and the angle between them" into a language the proving system will understand.

1.2 Project Aims

This project has two separate but interconnected goals; the production of a natural language processor, and the creation of a basic automated prover. The processor:

- Serves as a front end for any automated prover.
- Is usable without any understanding of symbolic notation or programming.
- Is modular enough to become the front-end to any automated proving system.
- Accurately translates standard English to symbolic notation.

The prover had far more modest aims:

- Understand basic questions in a limited scope, such as geometry or arithmetic.
- Take symbolic input, and return the truth value of said statement.
- Display the steps of the proof to the end user.

As the two pieces are separate, the processor is not limited by the prover, and may instead be joined with a more robust system by specifying what said system is. This is necessary to convert the request properly, so that the processor knows which symbolic language to translate to.

1.2.1 Natural Language Processor

The processor will takes a user's input in English, and determines the goal of the user based on the content of that input. The user is not required to know anything, other than that the system is intended for mathematical queries and will be able to do nothing else. Once the input is analyzed, the processor converts it to the proper symbolic language for the automated prover.

1.2.2 Automated Theorem Prover

The prover takes a mathematical claim, in a symbolic notation, and applies axioms, theorems, and other knowledge as necessary to prove or disprove the claim. If the prover is able to use theorems in its proofs, it is barred from using them in such a way as to give the appearance of circular logic. For example, when a user asks for a proof of the Pythagorean Theorem, the prover may not use the Pythagorean Theorem as its justification for the truth of the theorem. Should the user ask for a proof of an axiom, it is permissible to simply point to the axiom. The prover is not expected to explain an axiom to a user.

2 Previous Work

2.1 Natural Language Processors

Natural language processing has been an area of interest in computer science for the past 60 years. Initially, the subject was limited to machine translation, with the Georgetown experiment in 1954 being an early foray into the field. In this experiment over sixty Russian sentences were translated to English entirely automatically. The results were promising enough that the researchers anticipated automatic translation being solved within the next five years. [**Hutchins**] Progress was, obviously, much slower than this.

Funding was cut when the program ran over a decade, and progress in the field slowed. Two successful systems were created in the 1960s, SHRDLU and ELIZA.

2.1.1 SHRDLU

SHRDLU was a querying system which created a small “Blocks World” which was populated by cones, spheres, cubes, and other geometric shapes of various sizes and colors. [**winograd**] The user was able to instruct SHRDLU on how to move these objects around simply by specifying the shape in addition to its color or size. SHRDLU also had a simple memory system, allowing for reference to recently interacted with objects. This memory also allowed for SHRDLU to be queried on what she had previously done. SHRDLU’s world contained basic physics, allowing for the program to describe what was possible in the world and what was not. The final major feature was to remember the name a user gave an object or collection of objects. From this, it was possible to more easily instruct SHRDLU.

2.1.2 ELIZA

Include a sample dialogue, and a flowchart of ELIZA's function

2.1.3 Syntax and Semantics

2.1.4 Use in Information Retrieval

2.2 Automated Theorem Provers

2.2.1 Path Elimination

2.2.2 Domains of Knowledge

3 Software

3.1 Implementation Language

3.2 Natural Language Processor

3.2.1 Conversion to Transition Language

3.3 Automated Theorem Prover

4 Conclusions

4.1 Future Work