TJHSST Computer Systems Lab Senior Research Project French/English Translation 2006-2007

Sharon Ulery

January 23, 2007

Abstract

This project uses computational linguistics to serve students of French or English as a second language as well as those who know only one of these languages. The program will translate French to English and English to French well enough to be understandable to someone who knows only the output language. Even a less than perfect translation is useful for surfing the web, reading texts in a foreign language, and communication with someone from another country. It can also be used for students to check their writing by translating back to their native tongue. They can check mechanics and make sure the writing is comprehensible by checking these areas of the translation.

Keywords: computational linguistics, computer translation

1 Introduction - Elaboration on the problem statement, purpose, and project scope

1.1 Scope of Study

Starting with a word-for-word translation from French to English and vice versa, I will hard code grammar rules into the program so that it correctly

translates increasingly complex grammar structures. This project can change in size as needed throughout the year. At a minimum, the program should be able to deal with "subject verb object" type sentences with a wide vocabulary range in all tenses. At a maximum, the program will be able to translate all grammatically correct, non-idiomatic sentences in both languages with correct agreement of number and gender and context-specific translation of words with multiple definitions.

2 Background and review of current literature and research

Most current in this area is far above the introductory level of this research project. I read *Foundations of Statistical Natural Language Processing: Chapter 1* by Manning and Schutze. In this work, Manning and Schutze are considering the problem of having a computer "understand" natural language. They believe that language cannot be divided into "grammatical" and "ungrammatical" statements; rather there are more and less commonly used structures. They use a method such that the program learns the parts of speech of words and common syntactical structures by training it on a large body of input text from a wide variety of fields. Obviously, their methods do not produce perfect results, but this more modern approach is much more robust than the older approach of hardwiring all knowledge into the program at the beginning. It can be made to expand much more easily if the software grows by reading more text than if the programmers must write further grammar rules directly into the code. In fact, most current computer translation uses this type of statistical technique. Unfortunately, this approach is too sophisticated to learn and implement in a year-long project.

3 Development Pt. 1

The project will be considered successful if the output creates comprehensible output in the corresponding language from grammatically correct, sensical input.

I'm writing this program with the assumption that input will be about one to three sentences in length. It is then necessary to create output quickly enought that the user doesn't get impatient even if (s)he has many sentences

to input in succession. It also makes the job of translating harder, because there is little context with which to determine the meaning of the input. I used Java and the bilingual Hansard corpus, a set of parliamentary proceedings in both French and English from Canada. (Author's Note: I have not yet, but I probably will use this corpus.)

I used the Evolutionary Prototyping model for development. This means that I developed the system concept as I moved through the project. I chose this model because it is especially useful when requirements may change throughout the project and when it's not clear what the optimal architecture or algorithms to use are. I then assessed the project and chose to make changes when I saw results of tests or learned more about traditional methods in the field through my research.

I tested this program using the tried and true "eyeball it" method. I used specific structural testing to make sure that each new algorithm or increase in the sophistication of an algorithm worked the way I expected it to. I used functional testing to check that each "type" of sentence I could think of worked the way I expected it to. For example, a subject-verb-object sentence should translate using the subject to determine the conjugation of the verb. At the end of the project, I will ask a number of possible users to try it out using any sentences they can think of to input. The more grammatical and closer to literal the translation is, the better the code.

This project consists of two classes: Word and Driver. Word stores information about each wordtoken, including its content, stem, language, gender (if applicable), number (if applicable), verb type (if applicable), and part of speech. It includes methods to "get" and "set" each of these fields as well as methods to discover some of these attributes either from the dictionary or from the context of the word in the input phrase. The Driver class actually runs the program. It reads in the dictionaries, parses the input String, and calls all applicable methods of the Word class (including determineContent(), determineStem(), determineLang(), etc.).

See attached for visuals.