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

Sharon Ulery

June 5, 2007

**Abstract**

This project uses hard-coding techniques in the field of computational linguistics to translate French to English and English to French well enough to be understandable to someone who knows only the output language. The goal of this project was for the program to create comprehensible output in the corresponding language from grammatically correct, sensical input. Although the program has a very limited scope, this lack is due to insufficient resources, not poor technique. By expanding the program I created in a straightforward way, further grammar rules can simply be coded in, and the small dictionary I used can be easily expanded or replaced.

**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 gave the program grammar rules so that it correctly translates increasingly complex grammar structures. The translation mechanism uses only a small number of grammar rules because encoding extensive rules amounts to data entry. Using the same technique, however, the program could simply be 'taught' to translate all grammatically correct, non-idiomatic sentences in both languages with correct agreement of number and gender, assuming a perfect knowledge of each language's grammar.

## 1.2 Purpose

This project uses computational linguistics to serve students of French or English as a non-native language as well as those who know only one of these languages. The output should be understandable even 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.

## 2 Background and review of current literature and research

The field of computational linguistics originally used mostly word-for-word translation. As the field has developed, new techniques for parsing and translating natural language have been introduced. Eventually, various techniques relying on statistics have become the favored methods for natural language processing (NLP).

The advantage in using statistical techniques in NLP is that it is more robust: the program can grow and change as the material it works from grows and changes, and it doesn't require a division of language into "grammatical" and "ungrammatical" statements, which in English is an artificial boundary. Instead of having "grammatical" and "ungrammatical" statements, there are just syntactic structures that people are more or less likely to write.

Statistical translation techniques rely on a large bilingual corpus, which is a large text in both languages that was created using a human translator. The program then aligns sentences that say similar things in both languages and check to see how often a given word of the first language occurs when a given word of the second language appears in the matching sentence. The more often it happens, the more likely they are to translate each other.

Statistical techniques can be used at a variety of levels, from creating a dictionary as described above to learning about syntactic and even semantic organization at the level of paragraphs or even larger chunks of text. The program compares the frequency of various structures occuring in the translation language given various structures in the originial language to find how different grammar forms are best translated.

Unfortunately, using the full statistical techniques recommended by these sources is far beyond the scope of a year-long, high school project. I have, therefore, created a translation program using word-for-word translation.

# 3 Development

## 3.1 Approach

French and English are relatively similar in structure, since both are Romance languages that come from Latin, so a word-for-word translation is a successful basis for translation. This project accepts input and finds the translation of each word. It then outputs the results, changing the word order only based upon specific, hard-coded grammar rules. An example of such a rule would be if the input language were structured in subject - verb - object order and the output were structured in subject - object - verb order and the program swapped the order of every verb - object pair found in the input sentence. This project attempts to translate only between a pair of languages, so a direct translation system as described above is sufficient.

## 3.2 Components

This program was written in Java.

### 3.2.1 Dictionary

A bilingual dictionary is an essential part of any translation program. In fact, using two bilingual dictionaries is preferable, one from French to English and the other from English to French, such that the former is sorted based on the French input word and the latter is sorted based on the English input word.

Creating an extensive, correct dictionary is a project in and of itself. Given a bilingual corpus, it is possible to use statistical techniques to match words from the French text to words from the English text, but it is not simple to do so, and certainly not simple to do so well. An alternative method is to use an already created dictionary. I was unable to find a freely available French - English dictionary, so I chose to create my own "mini-dictionary" of about 30 words by hand. While this is clearly insufficient for a serious translation program, it includes all parts of speech and both genders of French words, so it is enough to demonstrate how the program works. If a larger dictionary were procured, it could easily replace the mini-dictionary to provide much more extensive translation capabilities.

### 3.2.2 Information Storage

The Word class has methods to determine and store information about any given word. Each input word is associated with a String of content, its root word in the form it is found in the dictionary, its gender (if applicable), its number, its part of speech, its verb type (irregular, -er verb, -ir verb, etc. – if applicable), and which language it is. This is all the information that the Driver uses to actually do translation.

### 3.2.3 Translation

The Driver class does the actual translation. It stores the input as Words, uses Word methods to determine all pertinent information about the input, and then prints to the screen the equivalent word in the output language in the appropriate order.

### 3.2.4 User Interaction

Users are guided through inputting the phrase to be translated and specifying whether the translation is French to English or English to French. At the beginning of the program, a GUI pops up that asks "What is the original language of the text? Quelle est la langue originale du texte? [the same text in French]" If the answer is written in English (i.e. "English" or "French"), a new GUI pops up that asks the user to "Please input the text." If the answer is written in French, the new GUI gives the same instructions in French. The intent here was to make the program as accessible to French-speakers as to English-speakers. If the user inputs something other than one of the languages the program can deal with, it simply outputs "I'm sorry, that is not a valid selection." in the terminal and quits the program.

Output is printed in the terminal. Output is in the form of a sentence in the target language. Punctuation within the sentence is not preserved, but there is a period at the end of the translation given.

## 3.3 Expansion

### 3.3.1 Technicalities

In writing this program, I experienced some difficulties in getting the necessary tools to make a useful system. The two most glaring technical problems involved were the lack of accents and the size of the dictionary.

The French language makes use of three main accents that can be placed on any vowel to change its pronunciation as well as the meaning of the word, along with a cidilla, which is an optional accent mark on a 'c', with the same effect. There is no way to simply input accents into the GUI interface into which users input phrases to be translated. In addition, the editor program I used did not allow accents either. Because of these difficulties, the program does not use accents at all and if a way were found to input them into the GUI, the words would not be found in the dictionary. This is a significant roadblock, however, because several common words differ only in their accent mark (e.g. "du" without an accent means "of the"; "du" with a circumflex over the u is the past participle of "to have to"). Even more importantly, the accent determines the tense of verbs between the present and the most common past tense. For the program to include conjugation and deconjugation techniques, it is important for it deal correctly with accents in both input and output. This is an important technical issue to address in for any expansion or improvement of the translator.

As explained in the Background, a bilingual dictionary is usually created using a large bilingual corpus and statistical techniques. Since I didn't choose to research this area, I was unable to create my own dictionary. Unfortunately, it is impossible to find a free or inexpensive online bilingual dictionary between French and English. For research purposes, I used a small bilingual dictionary of about thirty words, but the program's usability would be much enhanced if a larger dictionary were procured.

### 3.3.2 Finding Stems

A dictionary that included all correct forms of all words in the English language would be so large as to be completely impractical. Instead, the dictionary includes only the stems of words, such as "miss" for "misses" or "missed" and "girl" for "girls." Thus, to look up a word, it is necessary to find the word's stem.

The program determines a word's stem by going through the possibilities from most likely to least likely. Thus, it starts by checking whether the word is in the dictionary exactly as input. It then checks for whether it is a plural by dropping an 's' at the end, if there is one. In English, it then continues on to check 'ed' endings, among others and the equivalent endings in French.

There are quite a few common endings that need to be checked for when searching for word stems. Not all possible endings are included in the program as it stands, because there are too many to be useful as a teaching tool. It would be simple, however, to add new endings to look at when checking for stems. The methods for looking for stems are all included in the method determineStem() in Word and are set up as a long if – else if – then ladder.

### 3.3.3 Conjugation and Agreement

As the program stands at the end of the year, the program takes input, determines the stem words, and prints the translation of those stems. In any language, however, a given word is composed of a stem plus additional word parts (or morphemes) that make it correctly fit into the sentence. These additional morphemes may indicate tense, person, number, or (in French) gender of the word. When the program strips the word to its stem, it loses the information stored in the non-stem morphemes, so it can't retain that information in the translation. This means that the output is not conjugated for the correct person and tense, nor do adjectives agree in number and gender (if necessary) with the nouns they modify. Implementing conjugation and agreement capabilities would be an interesting and useful next step to improve this program.

# 4 Conclusion

## 4.1 Results

This project was deemed partially successful. Although the program has a very limited scope, this lack is due to insufficient resources, not poor technique.

The final version of the program determines the basic attributes of input words, including: gender, number, stem of root words and basic non-root forms or root words in the dictionary, and language. It then uses these attributes, combined with a bilingual dictionary, to output the translation of the input phrase.

Input is a text String of no more than 1000 words typed into a GUI. Each input word will be correctly dealt with if and only if it is in the mini-dictionary or is a simple form of a root word that is in the dictionary. The output is a word-for-word translation based on the roots of the input words, and is printed in the terminal. Information about each word, including its gender, number, stem, and language can be written in the terminal if the user desires. The form is: myContent + ": POS = " + myPOS + "; Verb Type = " + myVerbType + "; Gender = " + myGender + "; Number = " + myNumber + "; Stem = " + myStem + "; Language = " + myLanguage for each word. Information that is unknown about a Word is stored and output as "" or -1, for String and int fields respectively. Thus, if a stem cannot be determined, the stem is output as "".

## 4.2 Analysis

There are two methods of language processing that can be used in translation. Statistical analysis is used in most web-based and commercially available translation programs. Using that method, the translation program "learns" grammar rules by using a few basic rules and going over a large corpus of written material in both languages. This can be contrasted with the method I used, in which grammar rules are hard-coded into a program. The biggest obstacle in using hard-coded grammar rules is that to create a sufficient number of rules and a large dictionary requires a lot of data entry. As this wasn't beneficial in a senior tech lab project, the translation program I wrote is very primitive. However, by expanding the program I created in a straightforward way, further grammar rules can simply be coded in, and the small dictionary I used can be easily expanded or replaced. The program currently fits its specifications exactly, although those specifications are not extensive. If the program were extended through additional data entry, however, it would continue to exactly fulfill specifications as they grew.

# References

[1] Daniel Jurafsky and James M. Martin, "Words & Transducers: Morphology, Tokenization, Spelling", *Speech and Language Processing, 2nd Ed. 3*, 4 September 2006. http://www.cs.colorado.edu/ martin/slp2.html
This source is a good introduction to some of the main problems in artificially "understanding" natural language. It talks about parsing words into morphemes so that a dictionary can be a reasonable size and begins talking about understanding words in context.

[2] Daniel Jurafsky and James M. Martin, "Machine Translation", *Speech and Language Processing, 2nd Ed. 3*, 4 September 2006. http://www.cs.colorado.edu/ martin/slp2.html
This source discusses the problems involved in machine translation and gives an overview of how to address many of these issues.

[3] Kevin Knight, "Automating Knowledge Acquisition for Machine Translation".
This source is an overview of word-for-word statistics-based translation, syntax-based translation and semantics-based translation. It compared the usefulness of each and discussed high-level implementation techniques.

[4] Bonnie Webber, "Natural language processing", in AccessScience@McGraw-Hill, http://www.accessscience.com, DOI 10.1036/1097-8542.444850, last modified: April 10, 2000.