

# Assignment #5: Machine Translation

CSCI E-220 Artificial Intelligence

Due: Thursday, October 22, 2009

## Overview:

You will write a program in LISP (or in another programming language that you have confirmed the TA can read and grade) to translate sentences from English into French (or another language of your choice that uses the 26-letter Roman alphabet). The program will consist of three basic modules that work in series:

1. A recursive, depth-first-search parsing engine for parsing English sentences, with the language defined as a set of production rules.
2. A set of rules for modifying a structured English sentence into the structure employed by French (or your chosen language).
3. A dictionary lookup tool that turns a parsed sentence structure into a sentence in a new language, and prints or outputs the final sentence.

So, if your top-level call is

```
(translate '(the red cat sleeps) 'eng-to-fr)
```

then *translate* might simply be a call to your other three routines:

```
(lookup (modify (parse '(the red cat sleeps))) 'eng-to-fr) 'eng-to-fr)
```

and the final output would look something like

```
(LE CHAT ROUGE DORT) .
```

**You will DEMONSTRATE your algorithms using, at least, the following sentences:**

- **The red cat sleeps**
- **A happy woman walks to Cambridge**
- **Mary cut the table with a saw**
- **The woman reads a red book**
- **Mary and the woman saw the cane sugar**

## Part (1) – PARSE [70% credit]

Write a function (parse X) that takes a sentence expressed as a list of words, and parses it into a bottom-up tree, using a set of production rules and a lexicon. A suggested test set of production rules and a lexicon is included at the end of this assignment. Feel free to add on more test data.

DATA STRUCTURE: You can determine your own data structure for the output, but of course whatever it is will need to be handled by the next module. The structure of your program should be such that it is easy to add additional production rules and additional lexical symbols (words).

On input (PARSE (the red cat sleeps)) , you might produce output like

```
((V SLEEPS) (VP V) (N CAT) (ADJ RED) (DET THE) (NP DET ADJ N) (S NP VP))
```

which is the backward-directional implied-tree structure produced by the Tanimoto parser I demonstrated in class. It is easier to produce than a forward-directional nested tree, but may be harder to use as input to Modify and Lookup. You may instead prefer to provide a top-down nested tree structure such as:

```
((S (NP (DET THE) (ADJ RED) (N CAT)) (VP (V SLEEPS))))
```

or you can use any other representation as long as you make it clear how to read it.

PARSING ALGORITHM: You can use any parsing algorithm you like, but a depth-first search backtracking algorithm is likely to be the simplest, since it can be written recursively and requires minimal bookkeeping in terms of state information.

When using a depth-first search parsing algorithm and a self-recursive grammar such as the one for this assignment, you need to be sure that you have a termination condition on node expansion so that you do not recurse indefinitely down a single branch of the tree. One useful termination condition is to stop expanding a node if the number of leaves in the partial parse tree is greater than the number of input elements in your sentence – a final parse tree can never have fewer terminal symbols than it does non-terminal leaf nodes at any point during the parse.

For this assignment, do not worry about morphology, tense-case agreement and so forth (but see “optional extensions”, below).

## Part (2) – MODIFY [15% credit]

Write a function that takes the output from PARSE and a language-pair symbol

e.g. (MODIFY S ‘eng-to-french)

and modifies it using rules that explain how to turn English into French (or another language). For the purpose of this assignment I propose a single test rule, which is:

- Move adjectives from in front the nouns they modify to a place behind those nouns.

Again, the structure of your program and the mechanism for defining modification rules should be such that it is easy to plug in additional rules.

How exactly to apply that rule will depend on your data structure. If you have chosen to translate into a language other than French, you may need to develop and apply a different rule, of course – it should be one that is true about the difference between English and your language.

As an example, if your input to MODIFY is the output from PARSE above:

```
((V SLEEPS) (VP V) (N CAT) (ADJ RED) (DET THE) (NP DET ADJ N) (S NP VP))
```

then the output should be:

```
((V SLEEPS) (VP V) (ADJ RED) (N CAT) (DET THE) (NP DET N ADJ) (S NP VP))
```

### Part (3) – LOOKUP [15% credit]

Write a function that takes a modified parse tree, looks up the corresponding words in French (or your chosen language) in a lexicon grouped by parts of speech, and outputs the final sentence in French.

If your input to LOOKUP is the output from MODIFY, above:

((V SLEEPS) (VP V) (ADJ RED) (N CAT) (DET THE) (NP DET N ADJ) (S NP VP))  
then the final output should be something like:  
(LE CHAT ROUGE DORT)

A suggested lexicon of French words to use is provided at the end of this assignment.

#### Optional extensions:

**(a – up to 10%)** Add two-way translation, so that you can also translate from French back into English. While this should for the most part be relatively easy once you've done the assignment, there are some potential pitfalls, notably in the parsing of the word "a" which (if written without accents) is ambiguous as to whether it means "to" or is simply an auxiliary part of a past-tense verb. NOTE: If you are translating into a different language, then translating back out of that language into English may be easier, or harder, and may come with its own pitfalls to consider.

**(b – up to 20%)** Add some morphological analysis to your PARSE, MODIFY, and/or LOOKUP routines (wherever it makes sense) so that you can properly handle gender differences in languages with gender, or singular/plural issues, or past/present/future issues, or all three.

You will of course need to add a good deal more test data if you plan to attempt this part, so you would need some way of figuring out the gender, number, case, and tense of each noun and verb as an additional part of the state information, and ensure that like words match. In languages like English, with lots of overlapping use of the same word for different verb conjugations ("I see", "You see", "We see", "They see"), there may be some ambiguity that gets resolved only when translated into another language.

**(c – up to 30%)** Instead of using blind depth-first search, learn and code a more efficient parsing algorithm such as an Earley-algorithm Chart Parser.

A reminder that as with all optional extensions, these are totally open-ended. Do what you like with them, either right away or some time after you turn in the rest of the assignment. If you think you'd like to develop a final project on the topic of parsing, machine translation, or other areas in Natural Language Processing, let me know and we can discuss possibilities.

**APPENDIX: MINIMAL REQUIRED TEST DATA FOR ASSIGNMENT 5**  
(FEEL FREE TO EXPAND, OR TO CHANGE/ADD LANGUAGES)

English Language Production Rule Grammar:

S => NP VP | VP | S CONJ S

NP => DET N | PNOUN | DET ADJ N | NP PP | NP CONJ NP

VP => V | V NP | V PP

PP => PREP NP

English Lexicon:

N (nouns) : cat, woman, table, book, cane, saw, sugar

V (verbs) : sleeps, reads, walks, cut, saw

DET (determiners) : the, a

ADJ (adjectives) : red, happy, cane

CONJ (conjunctions) : and, or

PREP (prepositions) : with, to, from

PNOUN (proper noun): Mary, Cambridge

English to French modification rules:

ADJ N → N ADJ

English-to-French Translation Lexicon (by part of speech):

*(NOTES: treat any two-word phrases, e.g. “a vu” or “de canne”, as a single lexical symbol, and don’t worry about gender (Le vs. La) unless you are working on the optional morphology piece.)*

N (nouns) :

Cat → Chat

Woman → Femme

Table → Table

Book → Livre

Cane → Canne

Saw → Scie

Sugar → Sucre

ADJ (adjectives) :

Red → Rouge

Happy → Heureux

Cane → De Canne

CONJ (conjunctions) :

And → Et

Or → Ou

V (verbs) :

Sleeps → Dort

Reads → Lit

Walks → Marche

Cut → A coupé

Saw → A Vu

PREP (prepositions) :

With → Avec

To → A

From → De

DET (determiners) :

The → Le or La

A → Un

PNOUN (proper noun):

Mary

Cambridge