Implementation of NLP using prolog

P15194468

Sam Fuller

[Year]

# Simplification rules

The simplification rules are used for two purposes: to remove unnecessary rules, and to allow for synonyms to produce the same output. In my system I have added many simplification rules that allow more natural English to be used when consulting the system. For example: “show me all of the files on the c disk” will be shortened to “files on drive c”.

sr([X,drive|Y],[drive,X|Y]) :- char\_table(X,\_,\_).

One of the most important rules I added was to allow for “c drive” to be changed to “drive c”. In this rule it was necessary to use X instead of c because any letter can be used to label a drive. The rule makes use of the char\_table in et.pl to ensure that the drive letter is exactly one character long, if this check were not performed then entire words could be moved around drive, for example: “in drive c” becomes “drive in c”.

# Translation rules

The translation rules change the English keywords in to a msdos command that is sent to the operating system to be run. In addition to the rules given to us I have added multiple new rules with different functionality. Despite research I was unable to find a method to apply commands to the same command prompt window, therefore commands like cd (change directory) are irrelevant because a new window will open each time a command is issued.

## Name

tr([computer,called],['cmd /k hostname']).

tr([called,computer],['cmd /k hostname']).

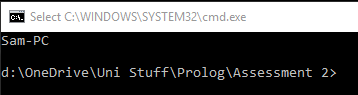
Example Inputs:

Command -> what is the name of this pc?

Command -> this computer is called ...

Command -> what is the computer named?

Output:



This command shows the name of the computer in a command prompt window. This was the first command I added and is therefore one of the simpler ones. This command has two translation rules to allow it to recognise many English inputs. The program only has one output, in this case ‘Sam-PC’ (I am testing the code on my home pc).

## Tree

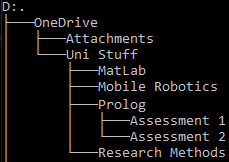
tr([tree,files,on,drive,X],['cmd /k cd / & cd /d ',X,': & tree']).

Example Inputs:

Command -> show me a diagram of the files on the d drive

Command -> what is the structure of the files on drive d

Output:



This command shows a tree like structure of all of the files on a given drive. Because ‘tree,files,on,drive,X’ also matches ‘y,files,on,drive,X’ this command must appear first, this will ensure that prolog tests the case against this translation rule first and therefore performs the correct action. Simplification rules have been used to allow for tree, diagram or structure to be used in addition to the drive letter rule.

The output shown here is a small part of the true output, the full tree contains sensitive file names and because it shows every folder in a drive is also too large to fit in this report.

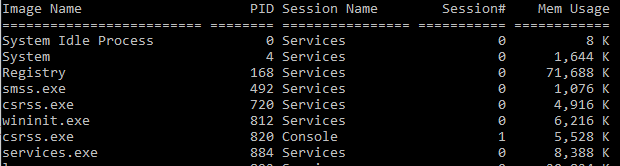
## Tasks

tr([all,tasks],['cmd /k tasklist']).

Example Inputs:

Command -> what current programs are running?

Output:



This command displays all current tasks the operating system is performing, again due to privacy and the size of the report the output has been limited in the amount of processes listed. This is yet another simple command, with a set input and a single output.

## Search

tr([search,X,on,drive,Y],['cmd /k cd / & cd /d ',Y,': & dir ',X,' /s /p']).

tr([search,X,files,on,drive,Y],['cmd /k cd / & cd /d ',Y,': & dir ',X,' /s /p']).

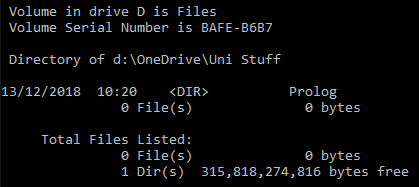
tr([search,files,called,X,on,drive,Y],['cmd /k cd / & cd /d ',Y,': & dir ',X,' /s /p']).

Example Inputs:

Command -> search for prolog in drive d

Command -> search for all files named prolog on the d drive

Outputs:



This command searches a given drive for a named file. This command has three variations to allow for many different inputs, however the command used is always the same. The output is a list of the directories that contain a ‘prolog’ directory, on my d drive there is only once folder named prolog.

The command sent to the operating system is the same in each variation, this command is split in to four parts:

* cmd /k Opens a new command prompt window.
* cd / Returns the directory to the drive mount point.
* cd /d ',Y,':Changes the directory to the correct drive.
* dir ',X,' /s /p Searches for all files with the correct name.

## Defrag

tr([defrag,drive,X],['cmd /k defrag ',X,': /U']).

tr([defrag,all,drives],['cmd /k defrag /C /U']).

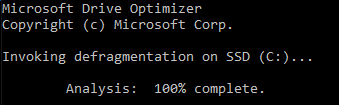
tr([defrag,all,drives,except,drive,X],['cmd /k defrag ',X,': /E /U']).

Example Inputs:

Command -> clean the c drive

Command -> defragment all of the drives

Outputs:





There are three different defrag commands that all do different things, one applies the command to a given drive, one applies the command to all drives and the last applies the command to all drives but a given drive. The command changes with by appending different suffixes, /u shows the current progress, /c applies the process to all drives and /e applies the process to all except the one listed.

Multiple simplification rules were created to allow for input of except/not/but/but not and defrag/defragment/clean.

It is important to note that because of the nature of the defrag msdos command, this only works if the system is launched as an administrator.

## Start

tr([start,X],['cmd /k start ',X]).

tr([start,X,called,Y],['cmd /k start ',X,' "',Y,'"']).

Examples:

Command -> open a text editor (opens text editor)

Command -> run the browser (opens internet explorer)

Command -> start matlab (opens matlab)

Command -> start notepad named helloworld (opens a file called helloworld, if one is not present it prompts for one to be created)

This command launches a program, it takes advantage of the fact that the start command looks for all .exe programs installed on the computer.

There are multiple simplification rules that help to simplify the input for notepad/text editor and iexplore/explorer/browser. These allow for more natural English (nobody uses iexplore in real life).

# Compare prolog to java

Before being taught this module I would have chosen to develop this program in java, simply because it is the language I am most familiar with.

In java a list of words to be removed and a list of substitutions could be implemented instead of the simplification rules, this part would be shorter and easier to understand. In addition, Java contains string manipulation libraries that include the string.split function which performs a similar task to the et.pl program. My java implantation would go through the list of inputted words and compare them to the lists of substitutions and apply any changes. Java streams would allow easy parallelisation of the task, however would make multiple word substitutions impossible, for example disk drive to drive. An alternative java solution might use the java regular expressions, applying a replace all function to replace disk drives with drives.

Since learning prolog, I now see the clear advantage it has in natural language processing over procedural languages. Coding in prolog is so convenient that there are java libraries that allow for a similar style of coding in java, although these libraries are written for the java style and therefore are less elegant.

# The Code

*%simplification rules ---------------------------------------------------*

sr([a|X],X).

sr([of|X],X).

sr([the|X],X).

sr([is|X],X).

sr([are|X],X).

sr([there|X],X).

sr([any|X],X).

sr([show|X],X).

sr([list|X],X).

sr([for|X],X).

sr([this|X],X).

sr([window|X],X).

sr([what|X],X).

sr([tell|X],X).

sr([me|X],X).

sr([about|X],X).

sr([find|X],X).

sr([program|X],X).

sr([running|X],X).

sr([in|X],[on|X]).

sr([pc|X],[computer|X]).

sr([stop|X],[shutdown|X]).

sr([closedown|X],[shutdown|X]).

sr([diagram|X],[tree|X]).

sr([structure|X],[tree|X]).

sr([back|X],[up|X]).

sr([operating,system|X],[about,system|X]).

sr([named|X],[called|X]).

sr([name|X],[called|X]).

sr([current|X],[all|X]).

sr([programs|X],[tasks|X]).

sr([disk,drive|X],[drive|X]).

sr([disk|X],[drive|X]).

sr([disk,drives|X],[drives|X]).

sr([disks|X],[drives|X]).

sr([X,drive|Y],[drive,X|Y]) :- char\_table(X,\_,\_).

sr([browser|X],[iexplore|X]).

sr([explorer|X],[iexplore|X]).

sr([text,editor|X],[notepad|X]).

sr([defragment|X],[defrag|X]).

sr([clean|X],[defrag|X]).

sr([but|X],[except|X]).

sr([not|X],[except|X]).

sr([but,not|X],[except|X]).

sr([run|X],[start|X]).

sr([open|X],[start|X]).

sr([new|X],[start|X]).

sr([folder|X],[files|X]).

sr([what,files|X],[files|X]).

sr([file|X],[files|X]).

sr([all,files|X],[files|X]).

sr([everything|X],[all,files|X]).

sr([every|X],[all|X]).

*%simplify ---------------------------------------------------------------*

simplify(List,Result) :-

sr(List,NewList),

!,

simplify(NewList,Result).

simplify([W|Words],[W|NewWords]) :-

simplify(Words,NewWords).

simplify([],[]).

*%translation rules ------------------------------------------------------*

tr([computer,called],['cmd /k hostname']).

tr([called,computer],['cmd /k hostname']).

tr([tree,files,on,drive,X],['cmd /k cd / & cd /d ',X,': & tree']).

tr([all,tasks],['cmd /k tasklist']).

tr([search,X,on,drive,Y],['cmd /k cd / & cd /d ',Y,': & dir ',X,' /s /p']).

tr([search,X,files,on,drive,Y],['cmd /k cd / & cd /d ',Y,': & dir ',X,' /s /p']).

tr([search,files,called,X,on,drive,Y],['cmd /k cd / & cd /d ',Y,': & dir ',X,' /s /p']).

tr([defrag,drive,X],['cmd /k defrag ',X,': /U']).

tr([defrag,all,drives],['cmd /k defrag /C /U']).

tr([defrag,all,drives,except,drive,X],['cmd /k defrag ',X,': /E /U']).

tr([start,X],['cmd /k start ',X]).

tr([start,X,called,Y],['cmd /k start ',X,' "',Y,'"']).

tr([quit],[quit]).

tr([files,on,drive,X],['cmd /k dir ',X,':']).

tr([X,files,on,drive,Y],['cmd /k dir ',Y,':\*.',X]).

tr([copy,files,from,drive,X,to,drive,Y], ['copy ',X,':\*.\* ',Y,':']).

tr([files,on,directory,X],['cmd /k dir \\',X]).

*%translate --------------------------------------------------------------*

translate(Input,Result) :-

tr(Input,Result),

!.

translate(\_,[]) :-

write('I do not understand'),

nl.

*%setup command ----------------------------------------------------------*

process\_commands :-

repeat,

write('Command -> '),

tokenize\_line(user,X),

tokens\_words(X,What),

simplify(What,SimplifiedWords),

translate(SimplifiedWords,Command),

pass\_to\_os(Command),

Command == [quit],

!.

*%passing commands to the OS ---------------------------------------------*

pass\_to\_os([]) :- !.

pass\_to\_os([quit]) :- !.

pass\_to\_os(Command) :-

concat(Command,String),

win\_exec(String,show).

concat([H|T],Result) :-

name(H,Hstring),

concat(T,Tstring),

append(Hstring,Tstring,Result).

concat([],[]).

append([H|T],L,[H|Rest]) :- append(T,L,Rest).

append([],L,L).