Implementation of NLP using Prolog

IMAT5118 - AI Programming - Assignment 2

Steven Smith - P13167208

12th January 2017 13:45-13:55

What is to come?

- Overview of the problem
- The changes made to the system
- Simplified Rules
- Translation Rules
- Results
- Comparison of Prolog vs C#
- Questions
- Demonstration

Overview of the problem

- Prolog
  - Developed in the 1970s for Logic Programming - Alain Colmerauer, Robert Kowalski, Maarten van Emden
  - Facts, Rules, Queries, Pattern matching, Search Spaces, Back Tracking, Unification
- NLP
- Efficient Tokenizer - et.pl
  - Michael Covington - Tokenizing text into characters and then words
- msdos.pl => os.pl
  - Support Linux and Mac OS, as well as Windows
  - 6 additional commands supported
- Comparison of Prolog with C#
  - C# version of the system - CSharp_Shell_NLP.cs

The changes made to the system

- Supporting Linux & Mac OS (POSIX)
  - current_os(-OS) where OS = windows or posix
  - tr(+OS,+SimplifiedWords,-Command,+OriginalWords) where OriginalWords for Original Case
  - find_in_list(+X,+List,-Result)
- No reliance on Efficient Tokenizer - et.pl
  - Overly complicated for the requirements
  - split_string/4, maplist/3 - calls string_lower/2 and atom_string/2
- Test Driven Development
  - test(windows, 'i would like you to remove the path please .\\ test1 ', 'rmdir .\\ test1 ').
- Calling the OS process on Windows and POSIX systems
  - pass_to_os/2 - win_exec/2 replaced with process_create/3 and shell/1
- C# version of the system - CSharp_Shell_NLP.cs
  - OO Classes, Regular expressions, String matching & manipulation, LINQ

Simplified Rules

Equivalent Phrases	Synonyms	Stop Phrases	Stop Words
directory to⇒directory	disk⇒drive	i would⇒	please⇒
disk in drive⇒drive	file⇒files	can i⇒	me⇒
disk in⇒drive	every⇒all	can you⇒	the⇒
what files⇒files	content⇒contents	could i⇒	is⇒
everything⇒all files	in⇒on	could you⇒	are⇒
any files⇒all files	create⇒make	would you⇒	a⇒
files contents⇒contents files	delete⇒remove	will you⇒	there⇒
to make⇒make	switch⇒change	give me⇒	these⇒
to remove⇒remove	bye⇒quit	like you to⇒	any⇒
to change⇒change	exit⇒quit	like to⇒	like⇒
to copy⇒copy	running⇒using	am i⇒	of⇒
to show⇒show	path⇒directory	i am⇒	see⇒
			list⇒
			show⇒
			tell⇒
			what⇒
			which⇒
			you⇒
			my⇒

Translation Rules

1. Rule 6 - View the contents of a file.
2. Rule 7 - Query the current directory/path.
3. Rule 8 - Change the current working directory to a new directory.
4. Rule 9 - Create a new directory.
5. Rule 10 - Remove an existing directory.
6. Rule 11 - Output which OS is the system being executed on.

No.	Rule	Windows	POSIX (X⇒ActualX, Y⇒ActualY)
1	quit		
2	all files on drive X	dir X:	
3	X files on drive Y	dir Y:*.X	
4	copy files from X to Y	copy X Y	cp ActualX ActualY
5	files on directory X	dir X	ls ActualX
6	contents files X	c:\windows\system32\more X	less ActualX
7	current directory	cd	pwd
8	change directory X	cd X	cd ActualX
9	make directory X	mkdir X	mkdir ActualX
10	remove directory X	rmdir X	rmdir ActualX
11	os using	ver	uname -a

Results

```
2 ?- execute_tests.
I do not understand: [this,test,fails]
Test Failed (windows): This test fails
[[this,test,fails],[this,test,fails],"","This test SHOULD FAIL"]
I do not understand: [this,test,finds,no,match]
Test Passed (windows): This test finds no match
I do not understand: [this,test,also,finds,no,match]
…
Test Passed (posix): i would like you to remove the directory ./test1
Test Passed (posix): remove path ./test1
Test Passed (posix): delete path ./test1
Test Passed (posix): would you remove path ./test1
Test Passed (posix): please delete path ./test1
Test Passed (posix): i would like you to remove the path ./test1
Test Passed (posix): what os am i running
Test Passed (posix): which os am i using
Test Passed (posix): tell me the os i am running
Test Passed (posix): show me the os i am using

171 Tests Executed,170 Tests Passed,1 Tests Failed.
```

- Limitations
  - ? at the end of a question
    - .\test? would match .\test1
  - Limited Simplified Rules
  - Spaces in filenames
    - Parsing text breaks on space

Comparison of Prolog versus C#

- Implementation
  - Ignoring Performance - Application is small, keyboard interaction
  - Taking input from the command line - **Draw**
  - Simplifying the input text - **Prolog** (elegant minimalist)
  - Mapping the simplified words to their command using Translation rules - **Prolog** (simpler)
  - Passing the command to the OS command shell - **Draw**
- Other considerations
  - Unification - **Prolog**
  - Backtracking vs Search algorithms - **Draw**
  - How do developers select languages?
    - at least 50% of the respondents, “open source libraries” -D, “extending existing code” -D, “already used in group” -C#, “personal familiarity” -C# and “team familiarity” -**C#**
- Tiobe Index Jan 2017 - C# 4% (4), Prolog 0.7% (30)
- IEEE Spectrum - 2016 Top Prog Langs - C# 87% (6), Prolog 26% (37)

Any questions?

Then on to the demo!