Implementation of NLP using Prolog

George Stavrinos
p15228718
Supervisor: David Elizondo
Centre for Computational Intelligence, Software Technology Research Laboratory
Faculty of Technology, De Montfort University, Leicester

January 3, 2017

1 Introduction

For the final assignment of this module we need to create an implementation of Natural Language Processing using Prolog, and then compare Prolog with another classic language of our choice.

The first part of this report is focused on the NLP system that I developed based on the provided code. This section is the biggest and most important part of the report, in which the NLP system's rules are analysed, along with its capabilities and limitations. In the final part of the first section I present some examples of the usage, as well as the final results of my experiments.

The second part of the report focuses on the advantages and disadvantages of Prolog in NLP applications compared to the C++ programming language.

In the conclusion I am going to present an overview of my experience and highlight the knowledge gained from this assignment.

2 The System

In this section the implemented NLP system is presented.

The most important part of a NLP system is its ruleset. The rules cannot stand on their own though. An effective word tokenizer is also needed so that we are able to isolate only the useful part of the human natural language.

In the next paragraphs of this section, we are going to discover the rationale and rules of the system along with how these were chosen. We are also going to have a look at the word tokenizer that was provided, and in the end we are going to see some examples of the system working, followed by the final results of the system.

2.1 Rationale and Rules

Choosing the rules for a NLP system is not a straight forward task, because human language is not restrictive, leaving humans the freedom to choose from an extremely big variety of words to express the same context.

Before starting programming, I had to set some specifications for my NLP system. The set of Linux commands that my system should be able to run based on humans' needs is presented below:

1. Shutdown the computer

2. Reboot the computer
3. Run the Firefox internet browser
4. Run the Dolphin file manager
5. Show the number of files in a directory
6. Show the size of a file or folder
7. Show files in a directory
8. Copy a file or folder in a directory
9. Move a file or folder in a directory
10. Remove a file or folder
11. Rename a file or folder
12. Show the differences between two files
13. Create a file or directory
14. Create a shortcut for a file or folder
15. Search for a file
16. Show current working directory
17. Change directory

2.1.1 The Tokenizer

The first thing that I had to make sure before starting implementing rules to my NLP system, was that the provided tokenizer was sufficient for my needs.

With some first rules just to test the tokenizer, it was clear that the tokenizer needed some modifications. The provided tokenizer is a very effective one for extracting information from every-day natural language, but in our application we needed something more specific: natural language regarding computer commands.

Three modifications were made to the initial implementation that expanded its capabilites:

- Extract information from filenames, without tokekizing mixed alphanumeric filenames e.g. the folder test1.
- Extract information from filenames, without tokenizing filenames that include the type of file e.g. song.mp3.
- Extract information from path urls, without assigning the "/" symbol as a special character e.g. the folder /home/george/prolog

The above modifications did not reduce the system's overall performance. In contradiction, they enhanced the system's versatility, by enabling it to extract knowledge from even more complicated input.

2.1.2 Rule implementation

When I was satisfied with the performance of the tokenizer, I moved on to the implementation of the rules, based on the set of commands described earlier in this section.

The first thing I did to create my base set of rules, was to try to type the commands I wanted the system to run, using natural language, without making my sentences simpler or script-like. Based on those first sentences, a small set of rules was ready to translate natural language, to Linux commands. Of course, this base set was not enough. Since each human being expresses themselves in slightly different ways, I needed samples from different humans. This is why I created a small form, that eight of my friend volunteered to help me with. The form had the format shown below:

"Imagine that you are interacting with "Samantha" the OS from the movie "Her". Basically, a very clever OS, that can understand what you say. For the sake of simplicity, imagine that you can only ask the OS to run one of the commands below. –Here I was listing the set of commands, but I won't do it again, since they can be seen in the beginning of this section.– Your task is, for each command, to ask the OS to run these commands in a natural way, like you would do with a human."

After collecting the forms, I realized that I had a lot of work to do to make the system versatile enough. I tried running their commands using my base set of rules, and almost none of them could be understood by the system. After enriching my ruleset, I asked the same users to interact with the system freely, without any forms. The system had a much better performance, but again, did not understand every command that was given. After observing all the given answers from the users, I modified my ruleset for a third time. Three completely new users interacted with the system, and were quite pleased with the results. We will see more on the results on the final paragraphs of this section.

2.1.3 Rules analysis

Now that I have presented the methodology I followed to implement the ruleset for the NLP system, I am going to present and analyze the rules themselves. To see all the simplification (simplified) rules and the translation rules, please turn to ANNEX A and ANNEX B respectively.

The simplification rules is the first "filter" that the users' input goes through after tokenization. Simplification (simplified) rules aim to transform complex sentences into predefined keywords that the system knows how to interpret. My simplification rules follow some specific guidelines:

- Remove words that add no knowledge e.g. is, are, to, in, on, for, etc.
- Transform all synonym words or phrases to a specific one e.g. "search" to "find", "look (for, which is removed)" to "find", "close the computer" to "shutdown", etc.

The translation rules are the second and final "filter" that the users' input goes through before it is translated into a Linux command. The translation rules are directly connected with simplification rules, since their input is the simplified user input. Translation rules aim to transform simplified sentences into Linux commands, in contradiction with the simplification rules that are not involved into the transformation of whole sentences. Quite often, different sentences are translated into the same Linux command. This is due to the already mentioned variety of ways for humans to express the same context. The main focus of translation rules is to find all possible combinations of the simplified sentences, and translate them to the corresponding command.

2.2 Examples of the system working

In this part of the section we are going to see some examples of the system's usage. Instead of presenting examples from different sessions, I am going to present two demonstration (not complete) sessions of the system working with two different test users.

```
?- process_commands.
Hello! I am a small Prolog program created by George Stavrinos!
I can:
- Shutdown the computer
- Reboot the computer
- Run the Firefox internet browser
- Run the Dolphin file manager
- Show the number of files in a directory
- Show the size of a file or folder
- Show files in a directory
- Copy a file or folder in a directory
- Move a file or folder in a directory
- Remove a file or folder
- Rename a file or folder
- See the differences between two files
- Create a file or directory
- Create a shortcut for a file or folder
- Search for a file
- Show current working directory
- Change directory
What do you want me to do? - I want you to search for mp4 files in /home/george
/home/george/Downloads/task.mp4
What do you want me to do? - Please show me the current folder
/home/george/DMU/AI_Programming/prolog/final_project/my_solution
What do you want me to do? - Could you show me the size of the folder test1?
12K
What do you want me to do? - List all files on folder test1
total 8
-rw-rw-r-- 1 george george   2 Δεκ   8 20:51 file.txt
-rw-rw-r-- 1 george george 111 Δεκ   8 22:17 index.html
What do you want me to do? - Please copy all mp3 files from test2 to test1!
What do you want me to do? - I want to see all files in folder test1
total 12
-rw-rw-r-- 1 george george   2 Δεκ   8 20:51 file.txt
-rw-rw-r-- 1 george george 111 Δεκ   8 22:17 index.html
-rw-rw-r-- 1 george george   2 Δεκ  15 01:52 song.mp3
What do you want me to do? - Remove test1/song.mp3
What do you want me to do? - Please, show me all files on folder test1
total 8
-rw-rw-r-- 1 george george   2 Δεκ   8 20:51 file.txt
-rw-rw-r-- 1 george george 111 Δεκ   8 22:17 index.html
What do you want me to do? - Exit
Goodbye!
true.

?-
```

```
?- process_commands.
Hello! I am a small Prolog program created by George Stavrinos!
I can:
- Shutdown the computer
- Reboot the computer
- Run the Firefox internet browser
- Run the Dolphin file manager
- Show the number of files in a directory
- Show the size of a file or folder
- Show files in a directory
- Copy a file or folder in a directory
- Move a file or folder in a directory
- Remove a file or folder
- Rename a file or folder
- See the differences between two files
- Create a file or directory
- Create a shortcut for a file or folder
- Search for a file
- Show current working directory
- Change directory
What do you want me to do? - I want to see the difference between test2/t1.txt and test2/t2.txt
1,2c1,3
< prolog programming rocks!
< (but is difficult!)
---
> prolog programming rocks and
> is not difficult!
>
\ No newline at end of file
What do you want me to do? - Change the name of the file test2/t1.txt to test2/text1.txt
What do you want me to do? - show me the foles of folder test2
Sorry, but I do not quite understand...
What do you want me to do? - show me the files of folder test2
total 12
-rw-rw-r-- 1 george george  2 Δεκ   8 20:53 song.mp3
-rw-rw-r-- 1 george george 49 Δεκ  15 01:59 t2.txt
-rw-rw-r-- 1 george george 48 Δεκ  15 01:58 text1.txt
What do you want me to do? - Open Mozilla, please
What do you want me to do? - Create a shortcut for the file test1/index.html named test.html in folder test2
What do you want me to do? - Please, list every file in folder test2
total 12
-rw-rw-r-- 1 george george  2 Δεκ   8 20:53 song.mp3
-rw-rw-r-- 1 george george 49 Δεκ  15 01:59 t2.txt
lrwxrwxrwx 1 george george 16 Δεκ  15 02:07 test.html -> test1/index.html
-rw-rw-r-- 1 george george 48 Δεκ  15 01:58 text1.txt
What do you want me to do? - close
Goodbye!
true.

?-
```

2.3 Results

For the results of my system I used three users that were newly introduced to the system. They were instructed to ask the system to execute each of the tasks in two different ways, using natural language.

The system's performance per command as well as the overall performance are shown in the table below.

Command	Commands Understood	Performance Percentage
Shutdown the computer	6/6	100%
Reboot the computer	6/6	100%
Run the Firefox internet browser	6/6	100%
Run the Dolphin file manager	6/6	100%
Show the number of files in a directory	6/6	100%
Show the size of a file or folder	6/6	100%
Show files in a directory	6/6	100%
Copy a file or folder in a directory	6/6	100%
Move a file or folder in a directory	6/6	100%
Remove a file or folder	6/6	100%
Rename a file or folder	6/6	100%
Show the differences between two files	6/6	100%
Create a file or directory	**5/6**	**83.3%**
Create a shortcut for a file or folder	**4/6**	**66.6%**
Search for a file	**3/6**	**50%**
Show current working directory	6/6	100%
Change directory	6/6	100%
Overall Performance	**96/102**	**94.1%**

The table above shows an effective NLP system that has some flaws. Most of the problems with my implementation have been already found, based on the commands that were not understood. The list below shows the most important problems with the current NLP system:

- No capital letters support: Two of the above six errors were encountered due to the fact that the user wanted to use a file o directory that included capital letters. Since Linux is case-sensitive with its files, although the system could understand the user's input, it could not generate the correct Linux command.
- Need for more flexibility: Three of the six problems came from user input that included the expected words, but did not have the expected syntax. For example, the user's input "Copy the file linux.pl to the test1 folder" although it contains all the expected words for a copy command, the name of the folder comes before the word "folder" making the system unable to understand this kind of wording.

3 Comparison of Prolog with a procedural language

In this section we are going to discuss the advantages and disadvantages of Prolog compared to C++ in NLP applications, based on my experience with the current and previous projects.

The Prolog programming language offers many advantages that cannot be found in other even more popular languages. Knowledge bases combined with the rule based syntax enable programmers to describe the logic behind a problem. In addition, the simple syntax of Prolog makes it easy to read, maintain and implement. Finally, Prolog offers built-in ways to manipulate lists and search on trees, making it suitable for NLP applications, that require searches on graphs.

Although Prolog has many advantages especially regarding NLP applications, it also has some disadvantages. The most major disadvantage of Prolog is its lack of native support from the Operating System. Prolog needs external tools in order to compile and run, making it inefficient compared to other languages that can run natively on the Operating System. Another disadvantage of Prolog is code debugging. Although we mentioned earlier that it is easy to implement tasks in Prolog, when bugs occur, it is very difficult to discover the root of the problem due to the backtracking feature of Prolog. Also, the lack of direct variable assignment and list editing, sometimes makes the code a bit more complicated than it would be in other programming languages.

The C++ programming language is one of the most widely used programming languages in the world. It offers many advantages compared to other languages, and this is the reason why it is so popular. An experienced C++ programmer can write extremely efficient code, and have full control over their code, from threads to garbage collecting. Also, the collection of external C++ libraries is almost infinite, making it a powerful tool for a wide range of problems, including NLP.

Although C++ is a powerful tool, reading, maintaining or developing code can be very frustrating even for a simple task. C++'s power comes combined with complexity, and this is why many programmers are turning their back to it: For the sake of simplicity. Developing code for a specific task may require a huge amount of code. Even with libraries that already implement some of the required tasks, coding remains cumbersome, since researching for and studying the desired library will most of the times consume a lot of time.

Prolog and C++ are two powerful languages. On the one hand, C++ is a general purpose language and on the other hand Prolog was specifically designed for NLP applications. Although C++ can solve almost any kind of problem, Prolog feels more effective and agile when it is used in NLP problems. This is due to the fact that Prolog has built-in functions that aim to simplify aspects of NLP-related problems. In addition, Prolog's descriptive syntax is ideal for NLP implementations. C++ might be one of the most popular programming languages, with the one of the richest library collections in the world, but falls short compared to Prolog when used in NLP problems. This is due to the fact the C++ can get quite complicated when too many libraries are combined together to simplify operations that Prolog can easily tackle (e.g. tree search).

Overall, Prolog is much more suitable for NLP problems, with its straight-forward rule-based syntax and its NLP-specific built-in functions, compared to C++ that although it can be used to solve any kind of problem, offers much less functions that can be used out-of-the-box in NLP.

4 Conclusion

A NLP system consists of many small modules that coordinated lead to the desired results. An effective tokenizer is very important, since every knowledge extraction operation that is executed is based on the tokenizer's results. Due to the fact that modeling a whole natural language is a very difficult and time consuming task, creating simplification and translation rules that balance the knowledge gained between too general and too specific is of vital importance.

Since NLP problems are already too complicated, programmers should not have to struggle (too much) with a programming language's particularities, but focus on the problem itself. This is where Prolog shines, due to its NLP-specific purpose. Powerful, yet quite simple, enables programmers to easily describe the context of their problem, and focus on its solution.

To conclude, working on a NLP system with Prolog is quite an enjoyable task, especially when watching your implementation interacting (successfully!) with real users. Such a task not only boosts a programmer's programming and problem-solving skills, but also enables them to think in alternate ways, in order to tackle seemingly complicated or even impossible NLP tasks.

ANNEX A – Simplification Rules

- sr([the|X],X).
- sr([is|X],X).
- sr([are|X],X).
- sr([there|X],X).
- sr([any|X],X).
- sr([everything|X],[all,files|X]).
- sr([every,file|X],[all,files|X]).
- sr([every,A,file|X],[all,A,files|X]).
- sr([i,want,you,to|X],X).
- sr([i,need,you,to|X],X).
- sr([what|X],X).
- sr([i|X],X).
- sr([want|X],X).
- sr([need|X],X).
- sr([see|X],X).
- sr([can|X],X).
- sr([could|X],X).
- sr([you|X],X).
- sr([yourself|X],X).
- sr([program|X],X).
- sr([a|X],X).
- sr([an|X],X).
- sr([of|X],X).
- sr([in|X],X).
- sr([on|X],X).
- sr([to|X],X).
- sr([would|X],X).
- sr([like|X],X).
- sr([for|X],X).
- sr([me|X],X).
- sr([my|X],X).
- sr([between|X],X).
- sr([under|X],X).
- sr([show|X],X).
- sr([tell|X],X).
- sr([print|X],X).
- sr([please|X],X).
- sr([new|X],X).
- sr([with|X],X).

- sr([make|X],[create|X]).
- sr([named|X],[name|X]).
- sr([called|X],[name|X]).
- sr([search|X],[find|X]).
- sr([look|X],[find|X]).
- sr([discover|X],[find|X]).
- sr([directory|X],[folder|X]).
- sr([delete|X],[remove|X]).
- sr([erase|X],[remove|X]).
- sr([wipe|X],[remove|X]).
- sr([transfer|X],[move|X]).
- sr([reboot|_],[reboot]).
- sr([restart|_],[reboot]).
- sr([restart,system|_],[reboot]).
- sr([restart,computer|_],[reboot]).
- sr([shutdown|_],[shutdown]).
- sr([power,off|_],[shutdown]).
- sr([power,off,system|_],[shutdown]).
- sr([power,off,the,system|_],[shutdown]).
- sr([power,off,computer|_],[shutdown]).
- sr([power,off,the,computer|_],[shutdown]).
- sr([close,the,computer|_],[shutdown]).
- sr([close,computer|_],[shutdown]).
- sr([close,the,pc|_],[shutdown]).
- sr([close,pc|_],[shutdown]).
- sr([how,many|X],[count|X]).
- sr([number|X],[count|X]).
- sr([compare|X],[difference|X]).
- sr([default,internet,browser|X],[firefox|X]).
- sr([default,browser|X],[firefox|X]).
- sr([mozilla,firefox|X],[firefox|X]).
- sr([firefox,internet,browser|X],[firefox|X]).
- sr([internet,browser|X],[firefox|X]).
- sr([browser|X],[firefox|X]).
- sr([mozilla|X],[firefox|X]).
- sr([default,file,manager|X],[dolphin|X]).
- sr([default,file,browser|X],[dolphin|X]).
- sr([file,manager|X],[dolphin|X]).
- sr([file,browser|X],[dolphin|X]).
- sr([open|X],[run|X]).

- sr([start|X],[run|X]).
- sr([shortcut|X],[symlink|X]).
- sr([symbolic,link|X],[symlink|X]).
- sr([change,the,name|X],[rename|X]).
- sr([change,name|X],[rename|X]).
- sr([current,working|X],[current|X]).
- sr([switch|X],[change|X]).

ANNEX B – Translation Rules

- tr([reboot],['sudo reboot']).
- tr([shutdown],['sudo shutdown -h now']).
- tr([run,firefox],['firefox &']).
- tr([firefox],['firefox &']).
- tr([run,dolphin],['dolphin &']).
- tr([dolphin],['dolphin &']).
- tr([count,files,folder,X],['ls ',X,' | wc -l']).
- tr([count,files,X],['ls ',X,' | wc -l']).
- tr([count,X,files,folder,Y],['ls ',Y,'/*.',X,' | wc -l']).
- tr([count,X,files,Y],['ls ',Y,'/*.',X,' | wc -l']).
- tr([files,folder,X],['ls -l ',X]).
- tr([files,X],['ls -l ',X]).
- tr([list,files,folder,X],['ls -l ',X]).
- tr([list,files,X],['ls -l ',X]).
- tr([all,files,folder,X],['ls -l ',X]).
- tr([all,files,X],['ls -l ',X]).
- tr([list,all,files,folder,X],['ls -l ',X]).
- tr([list,all,files,X],['ls -l ',X]).
- tr([size,folder,X],['du -cksh ',X,' | cut -f1 | tail -n1']).
- tr([size,file,X],['du -h ',X,' | cut -f1']).
- tr([size,X],['du -cksh ',X,' | cut -f1 | tail -n1']).
- tr([size,file,X,folder,Y],['du -h ',Y,'/',X,' | cut -f1']).
- tr([X,files,folder,Y],['ls -l ',Y,'/*.',X]).
- tr([list,X,files,folder,Y],['ls -l ',Y,'/*.',X]).
- tr([all,X,files,folder,Y],['ls -l ',Y,'/*.',X]).
- tr([all,X,files,Y],['ls -l ',Y,'/*.',X]).
- tr([list,all,X,files,folder,Y],['ls -l ',Y,'/*.',X]).
- tr([list,all,X,files,Y],['ls -l ',Y,'/*.',X]).
- tr([copy,files,from,folder,X,folder,Y], ['cp -r ',X,'/* ',Y]).
- tr([copy,files,from,X,Y], ['cp -r ',X,'/* ',Y]).
- tr([copy,all,files,from,folder,X,folder,Y], ['cp -r ',X,'/* ',Y]).

- tr([copy,all,files,from,X,Y], ['cp -r ',X,'/* ',Y]).
- tr([copy,file,X,folder,Y], ['cp ',X,' ',Y]).
- tr([copy,folder,X,folder,Y], ['cp -r ',X,' ',Y]).
- tr([copy,folder,X,Y], ['cp -r ',X,' ',Y]).
- tr([copy,X,folder,Y], ['cp -r ',X,' ',Y]).
- tr([copy,X,Y], ['cp -r ',X,' ',Y]).
- tr([copy,X,files,from,folder,Y,folder,Z], ['cp ',Y,'/*.',X,' ',Z]).
- tr([copy,X,files,from,Y,Z], ['cp ',Y,'/*.',X,' ',Z]).
- tr([copy,all,X,files,from,folder,Y,folder,Z], ['cp ',Y,'/*.',X,' ',Z]).
- tr([copy,all,X,files,from,Y,Z], ['cp ',Y,'/*.',X,' ',Z]).
- tr([move,files,from,folder,X,folder,Y], ['mv ',X,'/* ',Y]).
- tr([move,files,Y], ['mv * ',Y]).
- tr([move,all,files,from,folder,X,folder,Y], ['mv ',X,'/* ',Y]).
- tr([move,all,files,from,X,Y], ['mv ',X,'/* ',Y]).
- tr([move,files,from,X,Y], ['mv ',X,'/* ',Y]).
- tr([move,all,files,Y], ['mv * ',Y]).
- tr([move,X,files,from,folder,Y,folder,Z], ['mv ',Y,'/*.',X,' ',Z]).
- tr([move,all,X,files,from,folder,Y,folder,Z], ['mv ',Y,'/*.',X,' ',Z]).
- tr([move,all,X,files,folder,Z], ['mv *.',X,' ',Z]).
- tr([move,all,X,files,from,Y,Z], ['mv ',Y,'/*.',X,' ',Z]).
- tr([move,X,files,from,Y,Z], ['mv ',Y,'/*.',X,' ',Z]).
- tr([move,all,X,files,Z], ['mv *.',X,' ',Z]).
- tr([move,file,X,from,folder,Y,folder,Z], ['mv ',Y,'/',X,' ',Z]).
- tr([move,folder,X,from,folder,Y,folder,Z], ['mv ',Y,'/',X,' ',Z]).
- tr([move,X,from,folder,Y,folder,Z], ['mv ',Y,'/',X,' ',Z]).
- tr([move,file,X,from,Y,Z], ['mv ',Y,'/',X,' ',Z]).
- tr([move,folder,X,from,Y,Z], ['mv ',Y,'/',X,' ',Z]).
- tr([move,X,from,Y,Z], ['mv ',Y,'/',X,' ',Z]).
- tr([move,file,X,folder,Z], ['mv ',X,' ',Z]).
- tr([move,folder,X,folder,Z], ['mv ',X,' ',Z]).
- tr([move,file,X,Z], ['mv ',X,' ',Z]).
- tr([move,folder,X,Z], ['mv ',X,' ',Z]).
- tr([move,X,folder,Z], ['mv ',X,' ',Z]).
- tr([move,X,Z], ['mv ',X,' ',Z]).
- tr([remove,files,from,folder,X], ['rm -rf ',X,'/*']).
- tr([remove,files,from,X], ['rm -rf ',X,'/*']).
- tr([remove,all,files,from,folder,X], ['rm -rf ',X,'/*']).
- tr([remove,all,files,from,X], ['rm -rf ',X,'/*']).
- tr([remove,X,files,from,folder,Y], ['rm ',Y,'/*.',X]).
- tr([remove,X,files,from,Y], ['rm ',Y,'/*.',X]).

- tr([remove,all,X,files,from,folder,Y], ['rm ',Y,'/*.',X]).
- tr([remove,all,X,files,from,Y], ['rm ',Y,'/*.',X]).
- tr([remove,file,X,from,Y], ['rm ',Y,'/',X]).
- tr([remove,file,X,from,folder,Y], ['rm ',Y,'/',X]).
- tr([remove,folder,X,from,Y], ['rm -r',Y,'/',X]).
- tr([remove,folder,X,from,folder,Y], ['rm -r',Y,'/',X]).
- tr([remove,X,from,folder,Y], ['rm -r ',Y,'/',X]).
- tr([remove,X,from,Y], ['rm -r ',Y,'/',X]).
- tr([remove,file,X], ['rm ',X]).
- tr([remove,folder,X], ['rm -r ',X]).
- tr([remove,X], ['rm -r ',X]).
- tr([difference,files,X,and,Y], ['diff ',X,' ',Y]).
- tr([difference,file,X,and,file,Y], ['diff ',X,' ',Y]).
- tr([difference,file,X,and,Y], ['diff ',X,' ',Y]).
- tr([difference,X,and,file,Y], ['diff ',X,' ',Y]).
- tr([differences,files,X,and,Y], ['diff ',X,' ',Y]).
- tr([differences,file,X,and,file,Y], ['diff ',X,' ',Y]).
- tr([differences,file,X,and,Y], ['diff ',X,' ',Y]).
- tr([differences,X,and,file,Y], ['diff ',X,' ',Y]).
- tr([differences,X,and,Y], ['diff ',X,' ',Y]).
- tr([difference,X,and,Y], ['diff ',X,' ',Y]).
- tr([rename,from,X,Y], ['mv ',X,' ',Y]).
- tr([rename,file,X,Y], ['mv ',X,' ',Y]).
- tr([rename,folder,X,Y], ['mv ',X,' ',Y]).
- tr([rename,X,Y], ['mv ',X,' ',Y]).
- tr([create,folder,X,folder,Y], ['mkdir ',Y,'/',X]).
- tr([create,folder,Y,name,X], ['mkdir ',Y,'/',X]).
- tr([create,folder,name,X,folder,Y], ['mkdir ',Y,'/',X]).
- tr([create,folder,folder,Y,name,X], ['mkdir ',Y,'/',X]).
- tr([create,Y,folder,name,X], ['mkdir ',Y,'/',X]).
- tr([create,folder,name,X], ['mkdir ',X]).
- tr([create,folder,X], ['mkdir ',X]).
- tr([create,folder,X,Y], ['mkdir ',Y,'/',X]).
- tr([create,file,X,folder,Y], ['touch ',Y,'/',X]).
- tr([create,file,Y,name,X], ['touch ',Y,'/',X]).
- tr([create,file,name,X,folder,Y], ['touch ',Y,'/',X]).
- tr([create,file,folder,Y,name,X], ['touch ',Y,'/',X]).
- tr([create,Y,file,name,X], ['touch ',Y,'/',X]).
- tr([create,file,name,X], ['touch ',X]).
- tr([create,file,name,X,Y], ['touch ',Y,'/',X]).

- tr([create,file,X], ['touch ',X]).
- tr([create,file,X,Y], ['touch ',Y,'/',X]).
- tr([create,symlink,folder,X,name,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,folder,X,folder,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,folder,X,folder,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,folder,X,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,folder,X,folder,Y,name,Z], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,folder,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,folder,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,file,X,folder,Y,name,Z], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,file,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,file,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,file,X,name,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,file,X,folder,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,file,X,folder,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,X,folder,Y,name,Z], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,X,folder,Y,name,Z], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
- tr([create,symlink,X,name,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,X,folder,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,X,folder,Y], ['ln -s ',X,' ',Y]).
- tr([create,symlink,file,X,Y], ['ln -s ',X,' ',Y]).
- tr([find,all,files,name,X], ['find / -name ',X]).
- tr([find,all,files,name,X,folder,Y], ['find ',Y,' -name ',X]).
- tr([find,all,files,folder,Y,name,X], ['find ',Y,' -name ',X]).
- tr([find,all,files,Y,name,X], ['find ',Y,' -name ',X]).
- tr([find,files,name,X], ['find / -name ',X]).
- tr([find,files,name,X,folder,Y], ['find ',Y,' -name ',X]).
- tr([find,files,folder,Y,name,X], ['find ',Y,' -name ',X]).
- tr([find,files,Y,name,X], ['find ',Y,' -name ',X]).
- tr([find,file,name,X], ['find / -name ',X]).
- tr([find,file,name,X,folder,Y], ['find ',Y,' -name ',X]).
- tr([find,file,folder,Y,name,X], ['find ',Y,' -name ',X]).
- tr([find,file,Y,name,X], ['find ',Y,' -name ',X]).
- tr([find,all,files,name,X,Y], ['find ',Y,' -name ',X]).
- tr([find,file,name,X,Y], ['find ',Y,' -name ',X]).
- tr([find,files,name,X,Y], ['find ',Y,' -name ',X]).
- tr([find,X,folder,Y], ['find ',Y,' -name ',X]).
- tr([find,X], ['find / -name ',X]).

- tr([find,X,files,folder,Y],['find ',Y,' -name *.',X]).
- tr([find,X,files,Y],['find ',Y,' -name *.',X]).
- tr([find,X,files],['find / -name *.',X]).
- tr([find,all,X,files,folder,Y],['find ',Y,' -name *.',X]).
- tr([find,all,X,files,Y],['find ',Y,' -name *.',X]).
- tr([find,all,X,files],['find / -name *.',X]).
- tr([current,folder],['pwd']).
- tr([switch,folder,X],['cd ',X]).
- tr([switch,current,folder,X],['cd ',X]).
- tr([go,folder,X],['cd ',X]).
- tr([go,X],['cd ',X]).

ANNEX C – Tokenizer Prolog Code

```
% et.pl - M. Covington      2003 February 12

% ET the Efficient Tokenizer

% Measured speed: On a 1-GHz Pentium III,
% about 6 seconds per megabyte of text tokenized,
% or about three lines of text per millisecond.

% New in this version: Special handling of apostrophes.
% The apostrophe is a whitespace character except in
% the sequence 't which is treated as just t,
% making morphological analysis easier.


%%
%% User-callable routines
%%

% tokens_words(+Tokens,-Words)
%  From the output of the other routines, extracts just
%  the word tokens and converts them to atoms.

tokens_words([],[]).

tokens_words([w(Chars)|Tokens],[Atom|Atoms]) :-
   !,
   atom_chars(Atom,Chars),
   tokens_words(Tokens,Atoms).

tokens_words([_|Tokens],Atoms) :-
   % skip non-word tokens
   tokens_words(Tokens,Atoms).



% tokenize_file(+Filename,-Tokens)
%  Reads and entire file and tokenizes it.

tokenize_file(Filename,Tokens) :-
   open(Filename,read,Stream),
   tokenize_stream(Stream,Tokens),
   close(Stream).


% tokenize_stream(+Stream,-Tokens)
%  Reads an entire stream and tokenizes it.

tokenize_stream(Stream,[]) :-
   at_end_of_stream(Stream),
   !.

tokenize_stream(Stream,Tokens) :-
```

```
   tokenize_line_dl(Stream,Tokens/Tail),
   tokenize_stream(Stream,Tail).


% tokenize_line(+Stream,-Tokens)
%  Reads a line of input and returns a list of tokens.

tokenize_line(Stream,Tokens) :-
   tokenize_line_dl(Stream,Tokens/[]).


% tokenize_line_dl(+Stream,-Tokens/Tail)
%  Like tokenize_line, but uses a difference list.
%  This makes it easier to append the results of successive calls.

tokenize_line_dl(Stream,Tail/Tail) :-
   at_end_of_stream(Stream),                        % unnecessary test?
   !.

tokenize_line_dl(Stream,Dlist) :-
   get_char_and_type(Stream,Char,Type),
   tokenize_line_x(Type,Char,Stream,Dlist).


%%
%% Auxiliary predicates for tokenization
%%

% get_char_and_type(+Stream,-Char,-Type)
%  Reads a character, determines its type, and translates
%  it as specified in char_type_char.

get_char_and_type(Stream,Char,Type) :-
   get_char(Stream,C),
   char_type_char(C,Type,Char).


% tokenize_line_x(+Type,+Char,+Stream,-Tokens/Tail)
%  Tokenizes (the rest of) a line of input.
%  Type and Char describe the character that has been read ahead.

tokenize_line_x(eol,_,_,Tail/Tail) :-               % end of line mark; terminate
   !.

tokenize_line_x(whitespace,_,Stream,Dlist) :-       % whitespace, skip it
   !,
   tokenize_line_dl(Stream,Dlist).


% Word tokens and number tokens have to be completed,
% maintaining 1 character of read-ahead as this is done.
% NewChar and NewType are the character read ahead
% after completing the token.
```

```
tokenize_line_x(letter,Char,Stream,[w(T)|Tokens]/Tail) :-
   !,
   tokenize_letters(letter,Char,Stream,T,NewType,NewChar),
   tokenize_line_x(NewType,NewChar,Stream,Tokens/Tail).

tokenize_line_x(digit,Char,Stream,[n(T)|Tokens]/Tail) :-
   !,
   tokenize_digits(digit,Char,Stream,T,NewType,NewChar),
   tokenize_line_x(NewType,NewChar,Stream,Tokens/Tail).


% A period is handled like a digit if it is followed by a digit.
% This handles numbers that are written with the decimal point first.

tokenize_line_x(_, '.', Stream,Dlist) :-
   peek_char(Stream,P),
   char_type_char(P,digit,_),
   !,
   % Start over, classifying '.' as a digit
   tokenize_line_x(digit, '.', Stream,Dlist).

% Special characters and unidentified characters are easy:
% they stand by themselves, and the next token begins with
% the very next character.

tokenize_line_x(special,Char,Stream,[s(Char)|Tokens]/Tail) :-   % special char
   !,
   tokenize_line_dl(Stream,Tokens/Tail).

tokenize_line_x(_,Char,Stream,[other(Char)|Tokens]/Tail) :-     % unidentified char
   !,
   tokenize_line_dl(Stream,Tokens/Tail).



% tokenize_letters(+Type,+Char,+Stream,-Token,-NewChar,-NewType)
%   Completes a word token beginning with Char, which has
%   been read ahead and identified as type Type.
%   When the process ends, NewChar and NewType are the
%   character that was read ahead after the token.

tokenize_letters(letter,Char,Stream,[Char|Rest],NewType,NewChar) :-
   % It's a letter, so process it, read another character ahead, and recurse.
   !,
   get_char_and_type(Stream,Char2,Type2),
   tokenize_letters(Type2,Char2,Stream,Rest,NewType,NewChar).

tokenize_letters(digit,Char,Stream,[Char|Rest],NewType,NewChar) :-
   % It's a digit without space, so process it, read another character ahead, and recurse.
   !,
   get_char_and_type(Stream,Char2,Type2),
   tokenize_letters(Type2,Char2,Stream,Rest,NewType,NewChar).
```

```
tokenize_letters(_,'''',Stream,Rest,NewType,NewChar) :-
   %
   % Absorb an apostrophe, but only when it precedes t.
   % This keeps words together like doesn't, won't.
   %
   peek_char(Stream,t),
   !,
   get_char(Stream,_),
   tokenize_letters(letter,t,Stream,Rest,NewType,NewChar).

tokenize_letters(_, '.', Stream,['.'|Rest],NewType,NewChar) :-
   peek_char(Stream,P),
   char_type_char(P,letter,Char2),
   !,
   % It's a period followed by a letter, so include it and continue.
   % This is used to read string like this: filename.type
   get_char(Stream,_),
   tokenize_letters(letter,Char2,Stream,Rest,NewType,NewChar).

tokenize_letters(Type,Char,_,[],Type,Char).
   % It's not a letter, so don't process it; pass it to the calling procedure.


% tokenize_digits(+Type,+Char,+Stream,-Token,-NewChar,-NewType)
%   Like tokenize_letters, but completes a number token instead.
%   Additional subtleties for commas and decimal points.

tokenize_digits(digit,Char,Stream,[Char|Rest],NewType,NewChar) :-
   % It's a digit, so process it, read another character ahead, and recurse.
   !,
   get_char_and_type(Stream,Char2,Type2),
   tokenize_digits(Type2,Char2,Stream,Rest,NewType,NewChar).

tokenize_digits(_, ',', Stream,Rest,NewType,NewChar) :-
   peek_char(Stream,P),
   char_type_char(P,digit,Char2),
   !,
   % It's a comma followed by a digit, so skip it and continue.
   get_char(Stream,_),
   tokenize_digits(digit,Char2,Stream,Rest,NewType,NewChar).

tokenize_digits(_, '.', Stream,['.'|Rest],NewType,NewChar) :-
   peek_char(Stream,P),
   char_type_char(P,digit,Char2),
   !,
   % It's a period followed by a digit, so include it and continue.
   get_char(Stream,_),
   tokenize_digits(digit,Char2,Stream,Rest,NewType,NewChar).

tokenize_digits(Type,Char,_,[],Type,Char).
   % It's not any of those, so don't process it;
   % pass it to the calling procedure.
```

```
%%
%% Character classification
%%

% char_type_char(+Char,-Type,-TranslatedChar)
%   Classifies all characters as letter, digit, special, etc.,
%   and also translates each character into the character that
%   will represent it, converting upper to lower case.

char_type_char(Char,Type,Tr) :-
   char_table(Char,Type,Tr),
   !.

char_type_char(Char,special,Char).

% End of line marks
char_table(end_of_file, eol, end_of_file).
char_table('\n',        eol, '\n'       ).

% Whitespace characters
char_table(' ',     whitespace,  ' ').     % blank
char_table('\t',    whitespace,  ' ').     % tab
char_table('\r',    whitespace,  ' ').     % return
char_table('''',    whitespace, '''').     % apostrophe does not translate to blank

% Letters
char_table(a,     letter,    a ).
char_table(b,     letter,    b ).
char_table(c,     letter,    c ).
char_table(d,     letter,    d ).
char_table(e,     letter,    e ).
char_table(f,     letter,    f ).
char_table(g,     letter,    g ).
char_table(h,     letter,    h ).
char_table(i,     letter,    i ).
char_table(j,     letter,    j ).
char_table(k,     letter,    k ).
char_table(l,     letter,    l ).
char_table(m,     letter,    m ).
char_table(n,     letter,    n ).
char_table(o,     letter,    o ).
char_table(p,     letter,    p ).
char_table(q,     letter,    q ).
char_table(r,     letter,    r ).
char_table(s,     letter,    s ).
char_table(t,     letter,    t ).
char_table(u,     letter,    u ).
char_table(v,     letter,    v ).
```

```
char_table(w,     letter,    w ).
char_table(x,     letter,    x ).
char_table(y,     letter,    y ).
char_table(z,     letter,    z ).
char_table('A',   letter,    a ).
char_table('B',   letter,    b ).
char_table('C',   letter,    c ).
char_table('D',   letter,    d ).
char_table('E',   letter,    e ).
char_table('F',   letter,    f ).
char_table('G',   letter,    g ).
char_table('H',   letter,    h ).
char_table('I',   letter,    i ).
char_table('J',   letter,    j ).
char_table('K',   letter,    k ).
char_table('L',   letter,    l ).
char_table('M',   letter,    m ).
char_table('N',   letter,    n ).
char_table('O',   letter,    o ).
char_table('P',   letter,    p ).
char_table('Q',   letter,    q ).
char_table('R',   letter,    r ).
char_table('S',   letter,    s ).
char_table('T',   letter,    t ).
char_table('U',   letter,    u ).
char_table('V',   letter,    v ).
char_table('W',   letter,    w ).
char_table('X',   letter,    x ).
char_table('Y',   letter,    y ).
char_table('Z',   letter,    z ).
% Adding / as a letter so that I can read paths!
char_table('/',   letter,    / ).

% Digits
char_table('0',   digit,     '0' ).
char_table('1',   digit,     '1' ).
char_table('2',   digit,     '2' ).
char_table('3',   digit,     '3' ).
char_table('4',   digit,     '4' ).
char_table('5',   digit,     '5' ).
char_table('6',   digit,     '6' ).
char_table('7',   digit,     '7' ).
char_table('8',   digit,     '8' ).
char_table('9',   digit,     '9' ).

% Everything else is a special character.
```

ANNEX D – Main Prolog Code

```
sr([the|X],X).
sr([is|X],X).
sr([are|X],X).
sr([there|X],X).
sr([any|X],X).
sr([everything|X],[all,files|X]).
sr([every,file|X],[all,files|X]).


sr([every,A,file|X],[all,A,files|X]).
sr([i,want,you,to|X],X).
sr([i,need,you,to|X],X).
sr([what|X],X).
sr([i|X],X).
sr([want|X],X).
sr([need|X],X).
sr([see|X],X).
sr([can|X],X).
sr([could|X],X).
sr([you|X],X).
sr([yourself|X],X).
sr([program|X],X).
sr([a|X],X).
sr([an|X],X).
sr([of|X],X).
sr([in|X],X).
sr([on|X],X).
sr([to|X],X).
sr([would|X],X).
sr([like|X],X).
sr([for|X],X).
sr([me|X],X).
sr([my|X],X).
sr([between|X],X).
sr([under|X],X).
sr([show|X],X).
sr([tell|X],X).
sr([print|X],X).
sr([please|X],X).
sr([new|X],X).
sr([with|X],X).
sr([make|X],[create|X]).
sr([named|X],[name|X]).
sr([called|X],[name|X]).
sr([search|X],[find|X]).
sr([look|X],[find|X]).
sr([discover|X],[find|X]).
sr([directory|X],[folder|X]).
sr([delete|X],[remove|X]).
sr([erase|X],[remove|X]).
sr([wipe|X],[remove|X]).
sr([transfer|X],[move|X]).
```

```
sr([reboot|_],[reboot]).
sr([restart|_],[reboot]).
sr([restart,system|_],[reboot]).
sr([restart,computer|_],[reboot]).
sr([shutdown|_],[shutdown]).
sr([power,off|_],[shutdown]).
sr([power,off,system|_],[shutdown]).
sr([power,off,the,system|_],[shutdown]).
sr([power,off,computer|_],[shutdown]).
sr([power,off,the,computer|_],[shutdown]).
sr([close,the,computer|_],[shutdown]).
sr([close,computer|_],[shutdown]).
sr([close,the,pc|_],[shutdown]).
sr([close,pc|_],[shutdown]).
sr([how,many|X],[count|X]).
sr([number|X],[count|X]).
sr([compare|X],[difference|X]).
sr([default,internet,browser|X],[firefox|X]).
sr([default,browser|X],[firefox|X]).
sr([mozilla,firefox|X],[firefox|X]).
sr([firefox,internet,browser|X],[firefox|X]).
sr([internet,browser|X],[firefox|X]).
sr([browser|X],[firefox|X]).
sr([mozilla|X],[firefox|X]).
sr([default,file,manager|X],[dolphin|X]).
sr([default,file,browser|X],[dolphin|X]).
sr([file,manager|X],[dolphin|X]).
sr([file,browser|X],[dolphin|X]).
sr([open|X],[run|X]).
sr([start|X],[run|X]).
sr([shortcut|X],[symlink|X]).
sr([symbolic,link|X],[symlink|X]).
sr([change,the,name|X],[rename|X]).
sr([change,name|X],[rename|X]).
sr([current,working|X],[current|X]).
sr([switch|X],[change|X]).

simplify(List,Result) :-
  sr(List,NewList),
  !,
  simplify(NewList,Result).

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

simplify([],[]).

tr([quit],[exit]).
tr([exit],[exit]).
tr([close],[exit]).
tr([leave],[exit]).

% It seems that I won't need DCG!! :)
```

```
% Reboot
tr([reboot],['sudo reboot']).

% Shutdown
tr([shutdown],['sudo shutdown -h now']).

% Run Firefox
tr([run,firefox],['firefox &']).
tr([firefox],['firefox &']).

% Run Dophin File Browser
tr([run,dolphin],['dolphin &']).
tr([dolphin],['dolphin &']).

% Count all files in folder
tr([count,files,folder,X],['ls ',X,' | wc -l']).
tr([count,files,X],['ls ',X,' | wc -l']).

% Count type of files in folder
tr([count,X,files,folder,Y],['ls ',Y,'/*.',X,' | wc -l']).
tr([count,X,files,Y],['ls ',Y,'/*.',X,' | wc -l']).

% List all files in folder
tr([files,folder,X],['ls -l ',X]).
tr([files,X],['ls -l ',X]).
tr([list,files,folder,X],['ls -l ',X]).
tr([list,files,X],['ls -l ',X]).
tr([all,files,folder,X],['ls -l ',X]).
tr([all,files,X],['ls -l ',X]).
tr([list,all,files,folder,X],['ls -l ',X]).
tr([list,all,files,X],['ls -l ',X]).

% Size of folder
tr([size,folder,X],['du -cksh ',X,' | cut -f1 | tail -n1']).

% Size of file
tr([size,file,X],['du -h ',X,' | cut -f1']).

% Size of folder or file
tr([size,X],['du -cksh ',X,' | cut -f1 | tail -n1']).

% Size of file in folder
tr([size,file,X,folder,Y],['du -h ',Y,'/',X,' | cut -f1']).

% list type of files in folder
tr([X,files,folder,Y],['ls -l ',Y,'/*.',X]).
tr([list,X,files,folder,Y],['ls -l ',Y,'/*.',X]).
tr([all,X,files,folder,Y],['ls -l ',Y,'/*.',X]).
tr([all,X,files,Y],['ls -l ',Y,'/*.',X]).
tr([list,all,X,files,folder,Y],['ls -l ',Y,'/*.',X]).
tr([list,all,X,files,Y],['ls -l ',Y,'/*.',X]).
```

```
% Copy all files from folder to folder
tr([copy,files,from,folder,X,folder,Y], ['cp -r ',X,'/* ',Y]).
tr([copy,files,from,X,Y], ['cp -r ',X,'/* ',Y]).
tr([copy,all,files,from,folder,X,folder,Y], ['cp -r ',X,'/* ',Y]).
tr([copy,all,files,from,X,Y], ['cp -r ',X,'/* ',Y]).

% Copy specific file or folder to location
tr([copy,file,X,folder,Y], ['cp ',X,' ',Y]).
tr([copy,folder,X,folder,Y], ['cp -r ',X,' ',Y]).
tr([copy,folder,X,Y], ['cp -r ',X,' ',Y]).
tr([copy,X,folder,Y], ['cp -r ',X,' ',Y]).
tr([copy,X,Y], ['cp -r ',X,' ',Y]).

% Copy type of files from folder to folder
tr([copy,X,files,from,folder,Y,folder,Z], ['cp ',Y,'/*.',X,' ',Z]).
tr([copy,X,files,from,Y,Z], ['cp ',Y,'/*.',X,' ',Z]).
tr([copy,all,X,files,from,folder,Y,folder,Z], ['cp ',Y,'/*.',X,' ',Z]).
tr([copy,all,X,files,from,Y,Z], ['cp ',Y,'/*.',X,' ',Z]).

% Move all files from folder to folder
tr([move,files,from,folder,X,folder,Y], ['mv ',X,'/* ',Y]).
tr([move,files,Y], ['mv * ',Y]).
tr([move,all,files,from,folder,X,folder,Y], ['mv ',X,'/* ',Y]).
tr([move,all,files,from,X,Y], ['mv ',X,'/* ',Y]).
tr([move,files,from,X,Y], ['mv ',X,'/* ',Y]).
tr([move,all,files,Y], ['mv * ',Y]).

% Move type of files from folder to folder
tr([move,X,files,from,folder,Y,folder,Z], ['mv ',Y,'/*.',X,' ',Z]).
tr([move,all,X,files,from,folder,Y,folder,Z], ['mv ',Y,'/*.',X,' ',Z]).
tr([move,all,X,files,folder,Z], ['mv *.',X,' ',Z]).
tr([move,all,X,files,from,Y,Z], ['mv ',Y,'/*.',X,' ',Z]).
tr([move,X,files,from,Y,Z], ['mv ',Y,'/*.',X,' ',Z]).
tr([move,all,X,files,Z], ['mv *.',X,' ',Z]).

% Move specific file or folder from folder to folder
tr([move,file,X,from,folder,Y,folder,Z], ['mv ',Y,'/',X,' ',Z]).
tr([move,folder,X,from,folder,Y,folder,Z], ['mv ',Y,'/',X,' ',Z]).
tr([move,X,from,folder,Y,folder,Z], ['mv ',Y,'/',X,' ',Z]).
tr([move,file,X,from,Y,Z], ['mv ',Y,'/',X,' ',Z]).
tr([move,folder,X,from,Y,Z], ['mv ',Y,'/',X,' ',Z]).
tr([move,X,from,Y,Z], ['mv ',Y,'/',X,' ',Z]).
tr([move,file,X,folder,Z], ['mv ',X,' ',Z]).
tr([move,folder,X,folder,Z], ['mv ',X,' ',Z]).
tr([move,file,X,Z], ['mv ',X,' ',Z]).
tr([move,folder,X,Z], ['mv ',X,' ',Z]).
tr([move,X,folder,Z], ['mv ',X,' ',Z]).
tr([move,X,Z], ['mv ',X,' ',Z]).

% Remove all files from folder to folder
tr([remove,files,from,folder,X], ['rm -rf ',X,'/*']).
tr([remove,files,from,X], ['rm -rf ',X,'/*']).
tr([remove,all,files,from,folder,X], ['rm -rf ',X,'/*']).
```

```
tr([remove,all,files,from,X], ['rm -rf ',X,'/*']).

% Remove type of files from folder to folder
tr([remove,X,files,from,folder,Y], ['rm ',Y,'/*.',X]).
tr([remove,X,files,from,Y], ['rm ',Y,'/*.',X]).
tr([remove,all,X,files,from,folder,Y], ['rm ',Y,'/*.',X]).
tr([remove,all,X,files,from,Y], ['rm ',Y,'/*.',X]).

% Remove specific file or folder
tr([remove,file,X,from,Y], ['rm ',Y,'/',X]).
tr([remove,file,X,from,folder,Y], ['rm ',Y,'/',X]).
tr([remove,folder,X,from,Y], ['rm -r',Y,'/',X]).
tr([remove,folder,X,from,folder,Y], ['rm -r',Y,'/',X]).
tr([remove,X,from,folder,Y], ['rm -r ',Y,'/',X]).
tr([remove,X,from,Y], ['rm -r ',Y,'/',X]).
tr([remove,file,X], ['rm ',X]).
tr([remove,folder,X], ['rm -r ',X]).
tr([remove,X], ['rm -r ',X]).

% Differences between file X and file Y
tr([difference,files,X,and,Y], ['diff ',X,' ',Y]).
tr([difference,file,X,and,file,Y], ['diff ',X,' ',Y]).
tr([difference,file,X,and,Y], ['diff ',X,' ',Y]).
tr([difference,X,and,file,Y], ['diff ',X,' ',Y]).
tr([differences,files,X,and,Y], ['diff ',X,' ',Y]).
tr([differences,file,X,and,file,Y], ['diff ',X,' ',Y]).
tr([differences,file,X,and,Y], ['diff ',X,' ',Y]).
tr([differences,X,and,file,Y], ['diff ',X,' ',Y]).
tr([differences,X,and,Y], ['diff ',X,' ',Y]).
tr([difference,X,and,Y], ['diff ',X,' ',Y]).

% Rename file or filder from X to Y
tr([rename,from,X,Y], ['mv ',X,' ',Y]).
tr([rename,file,X,Y], ['mv ',X,' ',Y]).
tr([rename,folder,X,Y], ['mv ',X,' ',Y]).
tr([rename,X,Y], ['mv ',X,' ',Y]).

% Create new folder
tr([create,folder,X,folder,Y], ['mkdir ',Y,'/',X]).
tr([create,folder,Y,name,X], ['mkdir ',Y,'/',X]).
tr([create,folder,name,X,folder,Y], ['mkdir ',Y,'/',X]).
tr([create,folder,folder,Y,name,X], ['mkdir ',Y,'/',X]).
tr([create,Y,folder,name,X], ['mkdir ',Y,'/',X]).
tr([create,folder,name,X], ['mkdir ',X]).
tr([create,folder,X], ['mkdir ',X]).
tr([create,folder,X,Y], ['mkdir ',Y,'/',X]).

% Create new file
tr([create,file,X,folder,Y], ['touch ',Y,'/',X]).
tr([create,file,Y,name,X], ['touch ',Y,'/',X]).
tr([create,file,name,X,folder,Y], ['touch ',Y,'/',X]).
tr([create,file,folder,Y,name,X], ['touch ',Y,'/',X]).
tr([create,Y,file,name,X], ['touch ',Y,'/',X]).
```

```
tr([create,file,name,X], ['touch ',X]).
tr([create,file,name,X,Y], ['touch ',Y,'/',X]).
tr([create,file,X], ['touch ',X]).
tr([create,file,X,Y], ['touch ',Y,'/',X]).

% Create a symlink
tr([create,symlink,folder,X,name,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,folder,X,folder,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,folder,X,folder,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,folder,X,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,folder,X,folder,Y,name,Z], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,folder,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,folder,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,file,X,folder,Y,name,Z], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,file,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,file,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,file,X,name,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,file,X,folder,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,file,X,folder,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,X,folder,Y,name,Z], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,X,folder,Y,name,Z], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,X,name,Z,folder,Y], ['ln -s ',X,' ',Y,'/',Z]).
tr([create,symlink,X,name,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,X,folder,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,X,folder,Y], ['ln -s ',X,' ',Y]).
tr([create,symlink,file,X,Y], ['ln -s ',X,' ',Y]).
% I am not using the last two, because the phrases
% Create a shortcut for X in Y
% and
% Create a shortcut in Y for X
% have the same result!
%tr([create,symlink,folder,X,Y], ['ln -s ',X,' ',Y]).
%tr([create,symlink,X,Y], ['ln -s ',X,' ',Y]).


% Find files with name
tr([find,all,files,name,X], ['find / -name ',X]).
tr([find,all,files,name,X,folder,Y], ['find ',Y,' -name ',X]).
tr([find,all,files,folder,Y,name,X], ['find ',Y,' -name ',X]).
tr([find,all,files,Y,name,X], ['find ',Y,' -name ',X]).
tr([find,files,name,X], ['find / -name ',X]).
tr([find,files,name,X,folder,Y], ['find ',Y,' -name ',X]).
tr([find,files,folder,Y,name,X], ['find ',Y,' -name ',X]).
tr([find,files,Y,name,X], ['find ',Y,' -name ',X]).
tr([find,file,name,X], ['find / -name ',X]).
tr([find,file,name,X,folder,Y], ['find ',Y,' -name ',X]).
tr([find,file,folder,Y,name,X], ['find ',Y,' -name ',X]).
tr([find,file,Y,name,X], ['find ',Y,' -name ',X]).
tr([find,all,files,name,X,Y], ['find ',Y,' -name ',X]).
tr([find,file,name,X,Y], ['find ',Y,' -name ',X]).
tr([find,files,name,X,Y], ['find ',Y,' -name ',X]).
tr([find,X,folder,Y], ['find ',Y,' -name ',X]).
```

```
tr([find,X], ['find / -name ',X]).

% Find all files with type
tr([find,X,files,folder,Y],['find ',Y,' -name *.',X]).
tr([find,X,files,Y],['find ',Y,' -name *.',X]).
tr([find,X,files],['find / -name *.',X]).
tr([find,all,X,files,folder,Y],['find ',Y,' -name *.',X]).
tr([find,all,X,files,Y],['find ',Y,' -name *.',X]).
tr([find,all,X,files],['find / -name *.',X]).

% Show current working directory
tr([current,folder],['pwd']).

% Change directory
tr([switch,folder,X],['cd ',X]).
tr([switch,current,folder,X],['cd ',X]).
tr([go,folder,X],['cd ',X]).
tr([go,X],['cd ',X]).

translate(Input,Result) :-
   tr(Input,Result),
   !.

translate([],[]):-
   write('Oh, come on! Say something meaningful!'),
   nl,!.

translate(_,[]) :-
   write('Sorry, but I do not quite understand...'),
   nl.


process_commands :-
  write('Hello! I am a small Prolog program created by George Stavrinos! '),
  nl,
  write('I can: '),nl,
  write('- Shutdown the computer'),nl,
  write('- Reboot the computer'),nl,
  write('- Run the Firefox internet browser'),nl,
  write('- Run the Dolphin file manager'),nl,
  write('- Show the number of files in a directory'),nl,
  write('- Show the size of a file or folder'),nl,
  write('- Show files in a directory'),nl,
  write('- Copy a file or folder in a directory'),nl,
  write('- Move a file or folder in a directory'),nl,
  write('- Remove a file or folder'),nl,
  write('- Rename a file or folder'),nl,
  write('- See the differences between two files'),nl,
  write('- Create a file or directory'),nl,
  write('- Create a shortcut for a file or folder'),nl,
  write('- Search for a file'),nl,
  write('- Show current working directory'),nl,
  write('- Change directory'),nl,
```

```
   repeat,
      write('What do you want me to do? - '),
      tokenize_line(user,X),
      tokens_words(X,What),
      simplify(What,SimplifiedWords),
      translate(SimplifiedWords,Command),
      pass_to_os(Command),
      Command == [exit],
   !.

pass_to_os([])     :- !.

pass_to_os([exit]) :- write('Goodbye!'),!.

pass_to_os(Command) :-
   concat(Command,String),
   shell(String).


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).
```