AIP Assignment 2
NLP Implementation using SWI-Prolog

Dinis Marques Firmino - P13240786
Intelligent Systems MSc Faculty of Technology, De Montfort University, Leicester

January 19, 2018

1 Introduction

The focus of this assignment is on natural language processing (NLP) using SWI-Prolog. The main objective is to implement a NLP system that allows the user to interact with the operating systems' native command shell e.g. bash, using natural language queries. The second objective is to provide an in-depth comparison between a logic programming language such as Prolog, and a procedural language. The comparison should focus on the effectiveness, commodity, and performance of each language when solving NLP problems.

The reports' structure is as follows. Section 2 provides an overview of the implemented system, showing a list of all commands incorporated, additions to the rule-base such as simplification and translation rules, and necessary extensions to improve the scope of commands received by the system. Section 3 aims to demonstrate the end product by showing a few test session screen shots of the systems' functionality. Section 4 presents a discussion on the overall system performance, including what some of the limitations are, and what improvements could be made to enhance performance further and improve code structure. Finally, each stage of the project is summarised in the conclusion section.

2 System Overview

2.1 Commands

The first step in the development of the system was to choose a sensible number of Linux/Mac OS Bash Shell commands. The commands would ideally cover a range of system functionality such that the user can solely rely on the system and its NLP capabilities to interact with the operating system at ease.

Below is the list of the Mac OS/Linux (Bash) commands the system has implemented. Currently, there are a total of **16** commands, these are:

1. Shutdown the computer.
2. Launch applications (Google Chrome,)
3. Show current working directory.
4. File count in directory/folder.
5. Size of file or directory/folder.
6. Listing all files in a directory folder, as well as just listing files of certain extension type.
7. Copying all files from folder to folder, copying files of certain extension type, and copying single file/folder to another folder.

8. Moving all files from folder to folder, moving files of certain extension type, and moving single file/folder to another folder.

9. Removing all files from folder, removing files of certain extension type, and removing single files/folder.

2.2 Simplification Rules

The simplification rules are very important in dealing with the vague nature of language. This is because there is a large number of ways in which the same sentence can be articulated, and as such, a way to "narrow" down the words/tokens in a sentence to a single meaning is the core objective of the simplification rules. However, before this, the simplification rules should remove any words that are not meaningful to the context and thus don't aid the system in understanding the sentence. In NLP terminology, these words are called "stop words" and their removal is a very common pre-processing stage in NLP systems to reduce vocabulary size and improve overall performance. Then, to aid in combating the vagueness in language, the simplification rules should narrow down the synonyms of certain verbs and nouns into a single word so that the system is able to handle a variety of different words that effectively mean the same thing, and thus, should be translated to the same command.

To determine the simplification rules needed, one just has to imagine and write down the various possible combinations of english natural language sentences that translate to the same command. Then look for the key words that define the "action" for the system to perform. For example, when you want to delete a file you could say *"Is it possible to delete the file xyz.txt"*. In this case, the action would be *delete*, so the most appropriate synonyms of the word *delete* should be simplified such as *erase* and *eliminate*.

The simplification rules are attached in Appendix A.

2.3 Translation Rules

The translation rules are the next step after the simplification rules. They are responsible for mapping the simplified sentence to a specific bash shell command. To do this they must take into account the various combinations of tokens that are present in the simplified sentence after the simplified action in the sentence. Depending on the target command, some of these tokens will be variables, surrounded by key atoms. The end result is usually a variety of translation rules for the same output command to account for the variability.

The process for determining the translation rules for the user command *"Can you please show me everything in the folder desktop"* would be to first simplify the sentence using the above simplification rules which would result in the simplified sentence *"all files folder desktop"*. This simplified sentence would then be considered a translation rule for the command to display all files in the folder desktop. However, more translation rules need to be added to account for slightly different input commands such as if the word *"folder"* was after the variable name *"desktop"*, and another if there wasn't a word *"folder"* in the input command and the user was just referring to the *folder* solely by its name *"desktop"*.

The translation rules are attached in Appendix B

2.4 Extensions

While the core structure of the program remained intact from the provided project, there were some extensions that were crucial to improve the overall performance of the system. The necessity for these extensions emerged when performing tests on particular commands that required full file names with extensions, and paths. As it turns out the supplied sentence tokenizer [1] had some limitations with regards to how it dealt with these occurrences.

The latter problem was solved quickly by adding the character *'/'* as a letter to the char table in the et.pl file. This was not the best solution because *'/'* should actually be a special character, but it wasn't really an issue and in the end it worked well enough for this program. The program was now able to understand paths to directories.

The file name problem was slightly more complicated as it consisted of several different limitations. These limitation essentially prevented the program from reading file names. This was because the predicate in ET [1] called *Token Words* only returns full words comprised of groups of letters classified by the tokenizer. However, given the file name *"xyz.txt"* as input, the tokenizer would recognize *"xyz"* and *"txt"* as words, and "." as a special character. The predicate *Token Words* would then remove the special characters and the file name would be *"xyztxt"* which is wrong. To fix this problem, a new *tokenize letters* predicate was added to the tokenizer. This predicate essentially makes the tokenizer classify the special character *'.'* as a letter if it is followed by letters. The predicate is just the inverse of the predicate *"tokenize digits"* that adds support for decimal numbers. However, the newly added predicate does not override the functionality of the *tokenize digits* predicate, and therefore if a a number follows a '.', then the *'.'* is still classified as a special character, but since file extensions are always letters this isn't a problem. The predicate can be seen in Appendix C.1. The tokenizer performance before the extension can be seen in C.1.1, and the performance after in C.1.2.

Another problem with the tokenization of file names arose when alpha numeric names were introduced to the equation. A similar predicate to the above problem was added to the group of *tokenize letters* predicates. This time, the predicate was modified to read in digits and classify them as letters. The solution is not elegant, but it is better than modifying the character table to specify digits as letters, and still increases the systems performance when faced with most alphanumeric file names. The new predicate can be seen in Appendix C.2. The tokenizer performance before the extension can be seen in C.2.1, and the performance after in C.2.2.

3 Results

This section aims to discuss the results of the system and provide demonstrations of the system in action and its capabilities. The demonstration sessions will attempt to cover as many commands as possible.

3.1 Sessions

Appendix D shows **3** different sessions of the Prolog application being tested with a range of commands and its responses to each command.

3.2 Discussion

The sessions demonstrate that the system is very capable at performing a variety of tasks within the operating system. They also show that the system can handle a variety of natural language sentences for each command, albeit, it wasn't possible to show all the commands.

However, there are still significant limitations in the current implementation.

There is still much room for improvement in terms of adding more flexibility to the system. Flexibility is needed so that the system can interpret a wider variance of command sentences. Significant improvements can be made by adding many more simplification and translation rules. The latter is especially important because each command can be expressed in a variety of syntax structures. As such, the ability for the system to understand as many different ordering of words in a sentence for each and every command can be the difference between the system recognising a command and not.

Further extensions to the tokenizer would also be a desirable improvement to the current implementation. While the current extensions suffice for most cases, the system is still unable to recognise file names that start with numbers. The problem can be seen in Appendix E. The easy solution as mentioned above would be to convert numbers to letters in the character table, but that could be seen as a hard-coded solution so isn't really appropriate. A new predicate would likely be needed to resolve the problem.

The support for more commands should also be made a priority for possible improvements as the system only managed to cover 16. There are many more useful commands that the user could possibly need for appropriate interaction with the operating system.

Finally, the integration of a corpus database such as WordNet would be a huge bonus in both performance and reducing the number of simplification rules needed in the system. WordNet integrations are widely popular in Prolog application for NLP [2] because they provide a Prolog API that uses predicates to interface with *"synsets"* which could be used in the application to find the most appropriate synonyms of a word, and in turn eliminating the need to manually specific the simplification rules.

4 Comparison of Prolog with C++ (Declarative vs Procedural)

This section focuses entirely on the discussion of the various fundamental differences between logic and procedural programming. The languages that the discussion will compare will be Prolog and C++. The comparison should provide an in depth analysis of the advantages and disadvantages of each language when faced with a natural language processing problems, and how each fundamentally represent concepts and ideas in code.

Firstly, to compare Prolog with C++, the fundamental difference between the declarative and procedural programming paradigms must be analysed. The procedural paradigm is by far the most widely used in software. Related paradigms include Object-Oriented Programming (OOP) and Event-Driven Programming which contribute passively to the popularity of procedural paradigms. This paradigm favours a structured approach to programming logic with a clear set of steps to reach a desired goal. As such, the execution of such applications flows from top to bottom and ends. In contrast, the declarative paradigm, defines program logic as objects and facts that define the relationships between those objects. The use of rules is also predominant in the paradigm to assert facts within the domain rule-base. Unlike procedural paradigms, there is no clear direction of flow during execution.

C++ is one of the more widely used commercial programming languages. Its benefits make it a very suitable language for development of programs that need to operate at lower-levels and have high performance and low memory constraints. Examples of such programs/applications are drivers, embedded systems, games, video and sound processing engines, as well as many back-end software components that lay foundation to higher level components in applications.

The main benefits of C++ is that it provides a great deal of control over processes and memory operations which allows experienced developers to develop extremely efficient solutions to problems. This is due to the minimal abstraction of C++, and as such, requiring a much smaller stack of base code unlike higher level languages like Java and Python which require runtime that manage core functionality such as memory management. The language is also cross compatible and is built into most desktop operating systems. However, with increased control, comes increased complexity. C++ developers are usually the highest paid for a reason. This is because the required knowledge and time to write and maintain code is substantially higher than most other languages. It is often the case that C++ developers find themselves having to write many lines of code to achieve particular functionality, where in higher level languages 1-2 lines of code would suffice. This can make it particularly unsuited for most prototyping work. On top of this, knowledge on compilers, dynamic linking, etc..., is required to even set-up a program or use 3rd party libraries which can be frustrating process for people not familiar with the language.

Prolog is a very different language to C++, mainly because of the vastly different paradigm, but also in its purpose. The logic in a Prolog application is usually defined by a knowledge base which specifies objects and relationship constraints between them. The recursive mechanisms built into Prolog make it is very easy and efficient to backtrack through a large knowledge base and assert conditions, as long as the knowledge base is well defined beforehand. In practice, a programmer can describe a complex problem domain very quickly, while also maintaining good code readability.

However, simplicity is not always the case with Prolog. The paradigm and different programming concepts in themselves can be very alien to a programmer with a purely procedural background which can make the new way of thinking about problems difficult to grasp. This is not necessarily a disadvantage, but can in some cases outweigh the advantages due to the time it can take to adopt the language. The lack of a native IDE that provides an extensive suite of profiling and debugging tools is

one of the major disadvantages of Prolog, since they would be extremely useful to find the resolution to bugs in large knowledge bases which can be particularly hard because of the recursive nature of the Prolog. However, there are some predicates such as Trace that alleviate this issue somewhat.

While C++ is a desirable language for many problems, its use in the field of NLP is minimal when compared to other languages, especially Prolog. The backtracking nature of Prolog as discussed above, gives it the ability to very quickly traverse entire graphs and data structures of knowledge which is necessary as the complexity of the NLP problem grows. The equivalent could be achieved in C++, however, the code that would be required to rival the same level of performance achieved by Prolog at such tasks would take much longer to write and require a very experienced programmer. The lack of NLP libraries available for C++ as opposed to languages like Python makes this problem worse.

5 Conclusion

The implementation task in this assignment was beneficial to gain knowledge and understanding on the various design techniques and procedures involved in the development of a rule-based natural language processing system. From the many components required for a NLP system, it is reasonable to say that an effective tokenizer needs to be at the core of the functionality. It is of utmost importance for sentences to be broken down into individual tokens/words to make it easier for the system to extract the required knowledge.

The construction of the rule-base confirmed the complexity of natural language problems, even one of fairly narrow domain such as the one undertaken in this assignment. The problem is that modelling the vagueness and structural complexity of language using hand-crafted rules is a labour and time intensive task, especially in traditional languages. However, the power of Prolog helped soothe the complexity somewhat by directing the focus to be on the problem domain only. The simplicity in Prolog also saved a great deal of time as rules were able to be added, modified and removed with ease.

To conclude, the overall performance of the system was very acceptable, albeit with several limitations and improvements that could have been implemented into the final solution. The system was able to find a good balance between sentences written in natural language and script like commands by applying a generous amount of simplification and translation rules for each desired bash shell command. From both the practical and theoretical perspective of the coursework, it is clear that Prolog has been designed for NLP and there is no reason to doubt that the correct approach for any programmer tackling problems of similar nature is to adopt Prolog over most other traditional languages. The end product of this assignment is proof of this, as it shows that relatively little labour can achieve satisfying results that make it feel like one is naturally and intuitively interacting with a machine using language.

References

[1] Covington, M.A., and Georgia, U.S.A. (2003). *ET: an Efficient Tokenizer in ISO Prolog.* 18.

[2] Witzig, S., and Center, a. I.I. (2003). *Accessing wordnet from prolog.* Artificial Intelligence Centre, University of Georgia 118.

A Simplification Rules

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% START OF SIMPLIFICATION RULES %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sr([the|X],X).
sr([is|X],X).
sr([are|X],X).
sr([there|X],X).
sr([any|X],X).
sr([what|X],X).
sr([into|X],X).
sr([that|X],X).
sr([please|X],X).
sr([can|X],X).
sr([able|X],X).
sr([you|X],X).
sr([an|X],X).
sr([of|X],X).
sr([from|X],X).
sr([i|X],X).
sr([in|X],X).
sr([inside|X],X).
sr([within|X],X).
sr([we|X],X).
sr([to|X],X).
sr([my|X],X).
sr([mine|X],X).
sr([me|X],X).
sr([let|X],X).
sr([show|X], X).
sr([see|X], X).
sr([tell|X], X).
sr([place|X], X).
sr([put|X], X).
sr([now|X],X).
sr([possible|X],X).



% SR - Showing all files
sr([everything|X],[all,files|X]).
sr([all,content|X],[all,files|X]).
sr([the,content|X],[all,files|X]).
sr([content|X],[all,files|X]).
sr([all,the,files|X],[all,files|X]).
sr([each|X],[every|X]).



% Execute application
```

```
sr([launch|X],[execute|X]).
sr([run|X],[execute|X]).
sr([open|X],[execute|X]).
sr([start|X],[execute|X]).

% SR Web Browser


% SR Shutdown
sr([shutdown|_],[shutdown]).
sr([power,off|_],[shutdown]).
sr([power,off,system|_],[shutdown]).
sr([turn,off|_],[shutdown]).
sr([turn,off,system|_],[shutdown]).

% SR - Show current directory
sr([current,working|X], [current|X]).

% SR - Change
sr([change|X],[switch|X]).


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

% SR - Change the name of file or folder
sr([change,the,name|X], [rename|X]).
sr([change,the,name,of|X], [rename|X]).
sr([edit,the,name,of|X], [rename|X]).



% SR - Count files in directory/folder
sr([number|X], [count|X]).
sr([quantity|X], [count|X]).
sr([how,many|X], [count|X]).


% SR - Remove files
sr([delete|X], [remove|X]).
sr([erase|X], [remove|X]).
sr([eliminate|X], [remove|X]).


% SR - Move files
sr([carry|X], [move|X]).
sr([migrate|X], [move|X]).
sr([transfer|X], [move|X]).
sr([transport|X], [move|X]).


% SR - Copy files
sr([duplicate|X], [copy|X]).
```

B Translation Rules

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% START OF TRANSLATION RULES %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Exit Commands - Translation Rules %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% TR - System exiting

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Shutdown and Restart Commands - Translation Rules %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Switch and Show Directory Commands - Translation Rules %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

% TR - Switch current working directory
tr([switch,folder,X],['cd ',X]).
tr([switch,current,folder,X],['cd ',X]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% File Count Commands - Translation Rules %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% TR - File count in directory/folder

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% Execute Application Commands - Translation Rules %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% TR - Execute Google Chrome
```

```
tr([execute,chrome],['open -a "Google Chrome" --args']).
tr([execute,google,chrome],['open -a "Google Chrome" --args']).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% Listing Commands - Translation Rules %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% TR - List all files in folder

tr([all,files,folder,X],['ls -l ',X]).
tr([all,files,X,folder],['ls -l ',X]).
tr([all,files,X],['ls -l ',X]).
tr([every,file,folder,X],['ls -l ',X]).
tr([every,file,X,folder],['ls -l ',X]).
tr([every,file,X],['ls -l ',X]).
tr([list,all,files,folder,X],['ls -l ',X]).
tr([list,all,files,X,folder],['ls -l ',X]).
tr([list,all,files,X],['ls -l ',X]).
tr([list,every,file,folder,X],['ls -l ',X]).
tr([list,every,file,X,folder],['ls -l ',X]).
tr([list,every,file,X],['ls -l ',X]).

% TR - List all files of type in folder

tr([all,Y,files,folder,X],['ls -l ',X,'/*.',Y]).
tr([all,Y,files,X,folder],['ls -l ',X,'/*.',Y]).
tr([all,Y,files,X],['ls -l ',X,'/*.',Y]).
tr([every,Y,file,folder,X],['ls -l ',X,'/*.',Y]).
tr([every,Y,file,X,folder],['ls -l ',X,'/*.',Y]).
tr([every,Y,file,X],['ls -l ',X,'/*.',Y]).
tr([list,all,Y,files,folder,X],['ls -l ',X,'/*.',Y]).
tr([list,all,Y,files,X,folder],['ls -l ',X,'/*.',Y]).
tr([list,all,Y,files,X],['ls -l ',X,'/*.',Y]).
tr([list,Y,files,folder,X],['ls -l ',X,'/*.',Y]).
tr([list,Y,files,X,folder],['ls -l ',X,'/*.',Y]).
tr([list,Y,files,X],['ls -l ',X,'/*.',Y]).
tr([list,every,Y,file,folder,X],['ls -l ',X,'/*.',Y]).
tr([list,every,Y,file,X,folder],['ls -l ',X,'/*.',Y]).
tr([list,every,Y,file,X],['ls -l ',X,'/*.',Y]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% Copying Commands - Translation Rules %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% TR - Copy all files from folder/directory to folder/directory

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

```
tr([copy,all,files,X,Y], ['cp -r ',X,'/* ',Y]).
tr([copy,every,file,folder,X,folder,Y], ['cp -r ',X,'/* ',Y]).
tr([copy,every,file,X,folder,Y,folder], ['cp -r ',X,'/* ',Y]).
tr([copy,every,file,X,Y], ['cp -r ',X,'/* ',Y]).

% TR - Copy all files of type from folder/directory to folder/directory

tr([copy,Z,files,folder,X,folder,Y], ['cp ',X,'/*.',Z,' ',Y]).
tr([copy,Z,files,X,folder,Y,folder], ['cp ',X,'/*.',Z,' ',Y]).
%tr([copy,Z,files,X,Y], ['cp ',X,'/*.',Z,' ',Y]).
tr([copy,Z,files,folder,X], ['cp *.',Z,' ',X]).
tr([copy,Z,files,X,folder], ['cp *.',Z,' ',X]).
%tr([copy,Z,files,X], ['cp *.',Z,' ',X]).
tr([copy,all,Z,files,folder,X,folder,Y], ['cp ',X,'/*.',Z,' ',Y]).
tr([copy,all,Z,files,X,folder,Y,folder], ['cp ',X,'/*.',Z,' ',Y]).
%tr([copy,all,Z,files,X,Y], ['cp ',X,'/*.',Z,' ',Y]).
tr([copy,all,Z,files,folder,X], ['cp *.',Z,' ',X]).
tr([copy,all,Z,files,X,folder], ['cp *.',Z,' ',X]).
%tr([copy,all,Z,files,X], ['cp *.',Z,' ',X]).
tr([copy,every,Z,file,folder,X,folder,Y], ['cp ',X,'/*.',Z,' ',Y]).
tr([copy,every,Z,file,X,folder,Y,folder], ['cp ',X,'/*.',Z,' ',Y]).
%tr([copy,every,Z,file,X,Y], ['cp ',X,'/*.',Z,' ',Y]).
tr([copy,every,Z,file,folder,X], ['cp *.',Z,' ',X]).
tr([copy,every,Z,file,X,folder], ['cp *.',Z,' ',X]).
%tr([copy,every,Z,file,X], ['cp *.',Z,' ',X]).

% TR - Copy specific file/folder from folder/directory to folder/directory

tr([copy,file,Z,folder,X,folder,Y], ['mv ',X,'/',Z,' ',Y]).
tr([copy,file,Z,X,folder,Y,folder], ['mv ',X,'/',Z,' ',Y]).
tr([copy,file,Z,X,Y], ['mv ',X,'/',Z,' ',Y]).
tr([copy,folder,Z,folder,X,folder,Y], ['mv ',X,'/',Z,' ',Y]).
tr([copy,Z,folder,X,folder,Y,folder], ['mv ',X,'/',Z,' ',Y]).
tr([copy,folder,Z,X,Y], ['mv ',X,'/',Z,' ',Y]).
tr([copy,Z,folder,X,Y], ['mv ',X,'/',Z,' ',Y]).
tr([copy,file,Z,folder,X], ['mv ',Z,' ',X]).
tr([copy,file,Z,X,folder], ['mv ',Z,' ',X]).
tr([copy,file,Z,X], ['mv ',Z,' ',X]).
tr([copy,folder,Z,folder,X], ['mv ',Z,' ',X]).
tr([copy,Z,folder,X,folder], ['mv ',Z,' ',X]).
tr([copy,folder,Z,X], ['mv ',Z,' ',X]).
tr([copy,Z,folder,X], ['mv ',Z,' ',X]).
tr([copy,Z,X], ['mv ',Z,' ',X]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% Removing Commands - Translation Rules %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% - Remove files from folder/directory

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

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

% - Remove files of type from folder/directory

tr([remove,Y,files,folder,X], ['rm -rfi',X,'/*.',Y]).
tr([remove,Y,files,X,folder], ['rm -rfi',X,'/*.',Y]).
tr([remove,Y,files,X], ['rm -rfi',X,'/*.',Y]).
tr([remove,all,Y,files,folder,X], ['rm -rfi',X,'/*.',Y]).
tr([remove,all,Y,files,X,folder], ['rm -rfi',X,'/*.',Y]).
tr([remove,all,Y,files,X], ['rm -rfi',X,'/*.',Y]).
tr([remove,every,Y,file,folder,X], ['rm -rfi',X,'/*.',Y]).
tr([remove,every,Y,file,X,folder], ['rm -rfi',X,'/*.',Y]).
tr([remove,every,Y,file,X], ['rm -rfi',X,'/*.',Y]).

% TR - Remove specific file or folder from folder/directory

tr([remove,file,Y,folder,X], ['rm ',X,'/',Y]).
tr([remove,file,Y,X,folder], ['rm ',X,'/',Y]).
tr([remove,file,Y,X], ['rm ',X,'/',Y]).
tr([remove,folder,Y,folder,X], ['rm ',X,'/',Y]).
tr([remove,Y,folder,X,folder], ['rm ',X,'/',Y]).
tr([remove,folder,Y,X], ['rm -r',X,'/',Y]).
tr([remove,Y,folder,X], ['rm -r',X,'/',Y]).
tr([remove,Y,X], ['rm ',X,'/',Y]).
tr([remove,folder,Z], ['rm -rf',Z]).
tr([remove,Z,folder], ['rm -rf',Z]).
tr([remove,file,Z], ['rm ',Z]).
tr([remove,Z], ['rm ',Z]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% Moving Commands - Translation Rules %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% TR - Move all files from folder/directory to another folder/directory

tr([move,files,folder,X,folder,Y], ['mv ',X,'/* ',Y]).
tr([move,files,X,folder,Y,folder], ['mv ',X,'/* ',Y]).
tr([move,files,X,Y], ['mv ',X,'/* ',Y]).
tr([move,files,Y], ['mv * ',Y]).

tr([move,all,files,folder,X,folder,Y], ['mv ',X,'/* ',Y]).
tr([move,all,files,X,folder,Y,folder], ['mv ',X,'/* ',Y]).
tr([move,all,files,X,Y], ['mv ',X,'/* ',Y]).
tr([move,all,files,Y], ['mv * ',Y]).
tr([move,every,file,folder,X,folder,Y], ['mv ',X,'/* ',Y]).
tr([move,every,file,X,folder,Y,folder], ['mv ',X,'/* ',Y]).
tr([move,every,file,X,Y], ['mv ',X,'/* ',Y]).
tr([move,every,file,Y], ['mv * ',Y]).
```

```
% TR - Move files of type from folder/directory to another folder/directory

tr([move,Z,files,folder,X,folder,Y], ['mv ',X,'*.',Z,' ',Y]).
tr([move,Z,files,X,folder,Y,folder], ['mv ',X,'*.',Z,' ',Y]).
%tr([move,Z,files,X,Y], ['mv ',X,'*.',Z,' ',Y]).
tr([move,Z,files,X,folder], ['mv *.',Z,' ',X]).
tr([move,Z,files,folder,X], ['mv *.',Z,' ',X]).
%tr([move,Z,files,X], ['mv *.',Z,' ',X]).
tr([move,all,Z,files,folder,X,folder,Y], ['mv ',X,'*.',Z,' ',Y]).
tr([move,all,Z,files,X,folder,Y,folder], ['mv ',X,'*.',Z,' ',Y]).
%tr([move,all,Z,files,X,Y], ['mv ',X,'*.',Z,' ',Y]).
tr([move,all,Z,files,folder,X], ['mv *.',Z,' ',X]).
tr([move,all,Z,files,X,folder], ['mv *.',Z,' ',X]).
%tr([move,all,Z,files,X], ['mv *.',Z,' ',X]).
tr([move,every,Z,file,folder,X,folder,Y], ['mv ',X,'*.',Z,' ',Y]).
tr([move,every,Z,file,X,folder,Y,folder], ['mv ',X,'*.',Z,' ',Y]).
%tr([move,every,Z,file,X,Y], ['mv ',X,'*.',Z,' ',Y]).
tr([move,every,Z,file,folder,X], ['mv *.',Z,' ',X]).
tr([move,every,Z,file,X,folder], ['mv *.',Z,' ',X]).
%tr([move,every,Z,file,X], ['mv *.',Z,' ',X]).

% TR - Move specific file or folder from folder/directory to another folder/directory

tr([move,file,Z,folder,X,folder,Y], ['mv ',X,'/',Z,' ',Y]).
tr([move,file,Z,X,folder,Y,folder], ['mv ',X,'/',Z,' ',Y]).
tr([move,folder,Z,folder,X,folder,Y], ['mv ',X,'/',Z,' ',Y]).
tr([move,Z,folder,X,folder,Y,folder], ['mv ',X,'/',Z,' ',Y]).
tr([move,Z,folder,X,Y], ['mv ',X,'/',Z,' ',Y]).

tr([move,file,Z,X,folder], ['mv ',Z,' ',X]).
tr([move,file,Z,folder,X], ['mv ',Z,' ',X]).
tr([move,folder,Z,folder,X], ['mv ',Z,' ',X]).
tr([move,Z,folder,X,folder], ['mv ',Z,' ',X]).
tr([move,Z,folder,X], ['mv ',Z,' ',X]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% Renaming Commands - Translation Rules %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% Size Commands - Translation Rules %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

```
tr([how,big,folder,X],['du -cksh ',X,' | cut -f1 | tail -n1']).
tr([how,big,X,folder],['du -cksh ',X,' | cut -f1 | tail -n1']).
tr([size,file,X],['du -h ',X,' | cut -f1']).
tr([how,big,file,X],['du -h ',X,' | cut -f1']).
tr([size,X],['du -cksh ',X,' | cut -f1 | tail -n1']).
tr([how,big,X],['du -cksh ',X,' | cut -f1 | tail -n1']).
tr([size,file,X,folder,Y],['du -h ',Y,'/',X,' | cut -f1']).
tr([size,file,X,Y,folder],['du -h ',Y,'/',X,' | cut -f1']).
```

C ET Extensions

C.1 Fix Special Character '.'

```
% Copy and pasted from tokenize_digits section. Modified to ensure the
% tokenizer reads commands like "file.txt".
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.
get_char(Stream,_),
tokenize_letters(letter,Char2,Stream,Rest,NewType,NewChar).
```

C.1.1 Before Extension Performance

```
?- tokenize_line(user,X)
|    .
|: xyz.txt
X = [w([x, y, z]), s('.'), w([t, x, t])].

?-
```

Figure 1:

C.1.2 After Extension Performance

```
?- tokenize_line(user,X).
|: xyz.txt
X = [w([x, y, z, '.', t, x, t])].

?-
```

Figure 2:

C.2 Fix Alpha Numeric

```
% Modified to tokenize digits as words.
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).
```

C.2.1 Before Extension Performance

```
?- tokenize_line(user,X).
|: abc123xyz.txt
X = [w([a, b, c]), n(['1', '2', '3']), w([x, y, z]), s('.'), w([t, x, t])].
```

Figure 3:

C.2.2 After Extension Performance

```
?- tokenize_line(user,X).
|:  abc123xyz.txt
X = [w([a, b, c, '1', '2', '3', x|...])]
```

Figure 4:

D Example Sessions

D.1 Session 1

```
?- process_commands.
Hello? Is anyone there? I am a simple Prolog NLP application created by DMF
What do I do may you be asking?
At the moment I have been programmed to understand and translate to the following MacO
-- Shutdown the computer
-- Reboot the computer
-- Launch web browser
-- Show the current directory
-- Show files in a directory
-- Show the number of files in a directory
-- Show the size of a file or directory
-- Copy files
-- Move files
-- Delete files

What can I do for you? -
|: what is the current directory?
/Users/DYN/desktop/project

|: show me the content in /Users/DYN/desktop/project
total 120
drwxr-xr-x   2 DYN  staff     68 17 Jan 12:09 1folder
-rw-r--r--@  1 DYN  staff   2158 19 Jan 11:18 Example Commands
-rw-------   1 DYN  staff  16369 19 Jan 18:27 app.pl
-rw-r--r--   1 DYN  staff      0 17 Jan 19:47 audio.mp3
-rw-r--r--   1 DYN  staff      0 17 Jan 19:47 audio2.mp3
-rw-r--r--   1 DYN  staff      0 17 Jan 19:47 audio3.mp3
drwxr-xr-x   3 DYN  staff    102 19 Jan 18:26 desktop
drwxr-xr-x   4 DYN  staff    136 19 Jan 18:28 documents
-rw-------   1 DYN  staff   9937 19 Jan 09:57 et.pl
-rw-r--r--@  1 DYN  staff   9059 17 Jan 20:40 et_original.pl
drwxr-xr-x   6 DYN  staff    204 18 Jan 00:49 folder1
drwxr-xr-x   4 DYN  staff    136 19 Jan 16:29 folder2
-rw-r--r--   1 DYN  staff      0 19 Jan 16:04 phpfile.php
-rw-r--r--   1 DYN  staff      0 19 Jan 16:04 phpfile2.php
drwxr-xr-x  12 DYN  staff    408 19 Jan 16:49 resource
-rw-r--r--   1 DYN  staff      0 17 Jan 19:48 test.txt
-rw-r--r--   1 DYN  staff  14929 19 Jan 16:31 unix_old.pl

What can I do for you? -
|: how big is the documents folder
0B

What can I do for you? -
|: please migrate all mp3 files into the documents folder

|: please show me everything in the documents folder
total 0
-rw-r--r--  1 DYN  staff  0 17 Jan 19:47 audio.mp3
-rw-r--r--  1 DYN  staff  0 17 Jan 19:47 audio2.mp3
```

```
-rw-r--r--  1 DYN  staff  0 17 Jan 19:47 audio3.mp3
-rw-r--r--  1 DYN  staff  0 17 Jan 17:38 phpfile.php
-rw-r--r--  1 DYN  staff  0 17 Jan 17:38 phpfile2.php

|: can you duplicate every file in the documents folder into the folder1 folder

|: please show me all files in the folder folder1
total 0
-rw-r--r--  1 DYN  staff  0 19 Jan 18:41 audio.mp3
-rw-r--r--  1 DYN  staff  0 19 Jan 18:41 audio2.mp3
-rw-r--r--  1 DYN  staff  0 19 Jan 18:41 audio3.mp3
-rw-r--r--  1 DYN  staff  0 19 Jan 18:41 phpfile.php
-rw-r--r--  1 DYN  staff  0 19 Jan 18:41 phpfile2.php
-rw-r--r--  1 DYN  staff  0 17 Jan 19:48 test123.txt
-rw-r--r--  1 DYN  staff  0 16 Jan 21:51 textfile.txt

What can I do for you? -
|: what is the size of folder1?
0B

What can I do for you? -
|: how many files are there in folder1?
7

What can I do for you? -
|: please remove everything from folder1

What can I do for you? -
|: count the files in folder1?
0

What can I do for you? -
|: exit
true.

?-
```

D.2 Session 2

```
?- process_commands.
Hello? Is anyone there? I am a simple Prolog NLP application created by DMF
What do I do may you be asking?
At the moment I have been programmed to understand and translate to the following MacOs/Linux
-- Shutdown the computer
-- Reboot the computer
-- Launch web browser
-- Show the current directory
-- Show files in a directory
-- Show the number of files in a directory
-- Show the size of a file or directory
-- Copy files
-- Move files
-- Delete files
```

```
|: please list every file in /Users/DYN/Desktop/project/
total 120
drwxr-xr-x   2 DYN  staff     68 17 Jan 12:09 1folder
-rw-r--r--@  1 DYN  staff   2158 19 Jan 11:18 Example Commands
-rw-------   1 DYN  staff  16297 19 Jan 18:52 app.pl
drwxr-xr-x   3 DYN  staff    102 19 Jan 18:26 desktop
drwxr-xr-x   7 DYN  staff    238 19 Jan 18:30 documents
-rw-------   1 DYN  staff   9937 19 Jan 09:57 et.pl
-rw-r--r--@  1 DYN  staff   9059 17 Jan 20:40 et_original.pl
drwxr-xr-x   2 DYN  staff     68 19 Jan 18:45 folder1
drwxr-xr-x   4 DYN  staff    136 19 Jan 16:29 folder2
-rw-r--r--   1 DYN  staff      0 19 Jan 16:04 phpfile.php
-rw-r--r--   1 DYN  staff      0 19 Jan 16:04 phpfile2.php
drwxr-xr-x  12 DYN  staff    408 19 Jan 16:49 resource
-rw-r--r--   1 DYN  staff      0 17 Jan 19:48 test.txt
drwxr-xr-x   3 DYN  staff    102 19 Jan 18:47 testing
-rw-r--r--   1 DYN  staff  14929 19 Jan 16:31 unix_old.pl

|: can you rename the file test.txt to abc123xyz.txt

What can I do for you? -
|: can you rename the folder folder1 to home?

|: please can you transport the file abc123xyz.txt to the home folder

|: list everything in the home folder
total 0
-rw-r--r--  1 DYN  staff  0 17 Jan 19:48 abc123xyz.txt

What can I do for you? -
|: please copy all php files into the home folder
```

D.3 Session 3

```
|: quit
true.

?- process_commands.
Hello? Is anyone there? I am a simple Prolog NLP application created by DMF
What do I do may you be asking?
At the moment I have been programmed to understand and translate to the following MacOs/Linux
-- Shutdown the computer
-- Reboot the computer
-- Launch web browser
-- Show the current directory
-- Show files in a directory
-- Show the number of files in a directory
-- Show the size of a file or directory
-- Copy files
-- Move files
-- Delete files

What can I do for you? -
|: can you show me each file in the directory /Users/DYN/desktop/project
```

```
total 120
drwxr-xr-x   2 DYN  staff     68 17 Jan 12:09 1folder
-rw-r--r--@  1 DYN  staff   2158 19 Jan 11:18 Example Commands
-rw-------   1 DYN  staff  15428 19 Jan 22:53 app.pl
drwxr-xr-x   3 DYN  staff    102 19 Jan 18:26 desktop
drwxr-xr-x   7 DYN  staff    238 19 Jan 18:30 documents
-rw-------   1 DYN  staff   9937 19 Jan 09:57 et.pl
-rw-r--r--@  1 DYN  staff   9059 17 Jan 20:40 et_original.pl
drwxr-xr-x   4 DYN  staff    136 19 Jan 16:29 folder2
drwxr-xr-x   5 DYN  staff    170 19 Jan 19:21 home
-rw-r--r--   1 DYN  staff      0 19 Jan 16:04 phpfile2.php
drwxr-xr-x  12 DYN  staff    408 19 Jan 16:49 resource
drwxr-xr-x   2 DYN  staff     68 19 Jan 22:51 test1
drwxr-xr-x   7 DYN  staff    238 19 Jan 22:50 test2
drwxr-xr-x   4 DYN  staff    136 19 Jan 19:32 testing
-rw-r--r--   1 DYN  staff  14936 19 Jan 22:50 unix_old.pl

|: would you be able to show me the content within the home folder?
total 0
-rw-r--r--  1 DYN  staff  0 17 Jan 19:48 abc123xyz.txt
-rw-r--r--  1 DYN  staff  0 19 Jan 19:21 phpfile.php
-rw-r--r--  1 DYN  staff  0 19 Jan 19:21 phpfile2.php

What can I do for you? -
|: can you carry all files from the documents folder into the home folder?

What can I do for you? -
|: what are all the files in the home folder?
total 0
-rw-r--r--  1 DYN  staff  0 17 Jan 19:48 abc123xyz.txt
-rw-r--r--  1 DYN  staff  0 17 Jan 19:47 audio.mp3
-rw-r--r--  1 DYN  staff  0 17 Jan 19:47 audio2.mp3
-rw-r--r--  1 DYN  staff  0 17 Jan 19:47 audio3.mp3
-rw-r--r--  1 DYN  staff  0 17 Jan 17:38 phpfile.php
-rw-r--r--  1 DYN  staff  0 17 Jan 17:38 phpfile2.php

What can I do for you? -
|: please erase the file phpfile2.php
remove phpfile2.php? y

What can I do for you? -
|: quit
true.

?-
```

E Limitations

E.1 ET

```
?- tokenize_line(user,X).
|: 1abc.txt
X = [n(['1']), w([a, b, c, '.', t, x|...])].
```

Figure 5:

F Full Source Code