Asian Institute of Technology

School of Engineering and Technology

AT82.05 : Artificial Intelligence: Natural Language Understanding

Project Report

Smart Washing Machine

Submitted to

Prof. Phan Minh Dung

Submitted by

Chanapa Pananookooln (st121395)
Rafifa Islam (st121707)
Shin Thant (st121493)

Date of Submission

15/05/2021

Table of Contents

Topic

1. Background

In recent years, various smart devices have been equipped with advanced technologies. There are many functions in washing machines that can be used to wash clothes. As the functions of those devices become more complex, the complexity of running them has increased. However, complex operations may be carried out using simple natural language instructions. This will broaden the range of devices that can be operated by people who are not used to using them. Natural language understanding, on the other hand, is also a work in progress. Translating human-understandable language into machine-understandable language is extremely difficult. In our project, we suggested a method for controlling a washing machine using natural language commands. We proposed an effective interface using the simplest rule-based approach together with a dependency parser. Users would be able to wash using any cycle or option they want without having to understand all the features and machine control. We have followed the manual of a washing machine manufactured by Electrolux.

2. Related Works

2.1. NLP for Systems

Natural language has been studied as a means of informing and instructing robots. Users may define a task program that would be stored and executed by the robot. Their machine, like our dialog system, asks users to correct its (mis)understandings [6]. Their natural language comprehension, on the other hand, is based on keyword searches and assumes those words are spoken in a specific order. Another method allows a system to learn a series of behaviors from language instruction and dialog, as well as the lexical elements that apply to them [7]. Other researchers have used semantic parsing to make it easier for systems to learn the natural language [4]. Some researchers developed a dialog agent that interacts with users using natural language while learning semantic meanings from conversations [3]. Our system makes use of a more comprehensive, semantic understanding.

2.2. Chatbot

A chatbot device was considered to be the polar opposite of artificial intelligence, since it only provided pre-programmed responses to common questions stored in a database [9]. However, with the advent of numerous messenger applications such as Slack, Twitter, Facebook Messenger, and LINE, the chatbot system has become much more diversified and can now effectively serve as an interactive system for engaging in conversation [2]. Reservation systems in restaurants, hotels, and other facilities, as well as automated estimation generation, are examples of recent developments in chatbot systems [1]. We can see from these examples that a chatbot is more than just a responsive system; it is also an interactive interface that links users with other systems. In this paper we used parsing, this method entails analyzing the input text and manipulating it with a variety of natural language processing (NLP) functions. Centered on Combinatory Categorial Grammar (CCG), the authors of [5] and [8] represent meaning as an ontologically richly sorted, relational structure. We chose dependency parsing of the input sentence over logical approaches in our work because it is more convenient for knowledge extraction.

3. Methodology

Methodology is divided into two parts.

1. Grammar Creation
2. Translating Natural Language Instructions to Machine Understandable Language

3.1 Grammar Creation

The grammar creation process is divided into the following steps:

1. Create example instructions based on all possible actions
2. Parse example sentences with dependency parser
3. Group sentences with similar structures
4. Create a precise grammar based on the different groups

3.1.1. Instructions

We have created more than 100 example instructions based on all possible actions of the washing machine. Our work focused on a real manual from the Electrolux 24" Compact Washer with LuxCare Wash System - 2.4 Cu. Ft. which is a front load washing machine. In our reference menu (Figure 1), there are three types of situations: available options (ticked boxes), not available options (blank boxes), and only available options (black boxes).

Manual

	normal	activewear	colors	fast wash	delicates	hand wash	my favorite	clean washer	steam refresh	rinse & spin	jeans	sanitize	heavy duty	whitest whites
temperature														
sanitize												✓		
hot	✓								✓				✓	✓
warm	✓	✓	✓	✓	✓	✓							✓	✓
cold	✓	✓	✓	✓	✓						✓		✓	✓
tap cold	✓	✓	✓		✓					✓			✓	✓
spin speed														
max	✓		✓	✓						✓		✓	✓	✓
high	✓	✓	✓							✓		✓	✓	✓
medium	✓	✓	✓		✓					✓	✓			✓
low	✓	✓	✓		✓	✓				✓				
no spin					✓				✓	✓				
soil level														
max	✓		✓									✓	✓	
heavy	✓		✓									✓	✓	
medium/ normal	✓	✓	✓		✓	✓					✓	✓	✓	✓
light	✓	✓	✓	✓	✓									✓
extra light	✓	✓	✓	✓	✓									✓
options														
fresh rinse	✓	✓	✓		✓					✓		✓	✓	✓
prewash	✓												✓	✓
wrinkle release	✓	✓	✓		✓						✓	✓	✓	✓
control lock	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
mute sound	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
delay	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓

NO OPTIONS AVAILABLE WITH THIS CYCLE. DO NOT LOAD ANY ITEMS IN DRUM DURING CLEAN WASHER CYCLE.

✓ = Available selections. ▭ = Cycle defaults. ■ = Non-modifiable presets.

Figure 1. Reference menu of a washing machine produced by Electrolux

Figure 2. shows the available settings from the manual.

There are 13 possible ‘Cycles’ :

- normal
- activewear
- colors
- fast wash
- delicates
- hand wash
- my favorite
- steam refresh
- rinse & spin
- jeans
- sanitize
- heavy duty
- whitest white

*we did not implement the clean washer cycle

For each ‘Cycle’ there are 3 **‘functions’**, namely, ‘temperature’, ‘spin speed’ and ‘soil level’. There are also **‘extra options’** which include ‘fresh rinse’, ‘prewash’, wrinkle release’, ‘control lock’ and ‘mute sound’.

In each ‘cycle’ the available **‘selections’** for each ‘functions’ and the available ‘extra options’ are different. The available ‘selections’ and ‘extra options’ for each cycle are marked with a check mark. The grey-colored cells are the default ‘Selections’ for that ‘cycle’, while the black-colored cells with check marks indicate the only ‘Selection’ available for that ‘cycle’. The empty cells indicate that that specific ‘selection’ or ‘extra option’ is not available for that particular ‘cycle’.

For example, in the ‘delicates’ cycle the default ‘temperature’ is ‘warm’ but the users have the choice to change it to ‘cold’ or ‘tap cold’ but not ‘hot’ or ‘sanitize’ and the ‘prewash’ is not available for this cycle.

In ‘hand wash’ cycle the only ‘selection’ available for each ‘function’ are ‘warm’ ‘temperature’, The ‘delay’ option allows the user to delay the time that the washing machine starts working.

Notes:

- For ‘my favorite’ cycle the users can set their favorite ‘selection’ for each ‘function’ and save it such that the next time they choose ‘my favorite’ cycle the previously set combination would be performed. However, in this project we did not implement this part so we assume that when the user chooses the ‘my favorite’ cycle the default ‘selections’ for the ‘normal’ cycle would be performed.
- In ‘steam refresh’ and ‘rinse & spin’ cycle the spin speed is ‘None’.

Possible example of the instructions:

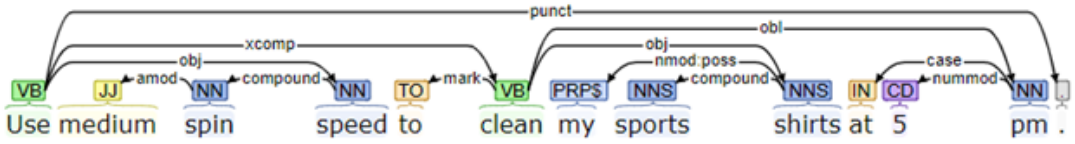
- Clean the clothes with the color cycle. Use tap cold water.
- Sanitize the clothes. Do not make any noise.
- Use hot water to clean the heavily dirty clothes.

3.1.2. Parsing

For this project we used a dependency parser from the python Stanza library. In dependency parser, the syntactic structure of a sentence is described solely in terms of the words (or lemmas) in a sentence and an associated set of directed binary grammatical relations that hold among the words.

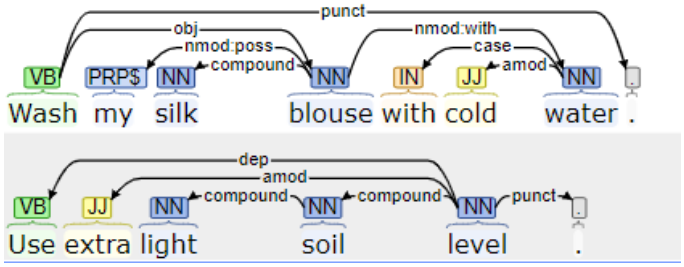
Instruction: Use medium spin speed to clean my sports shirts at 5 pm.

After parsing:



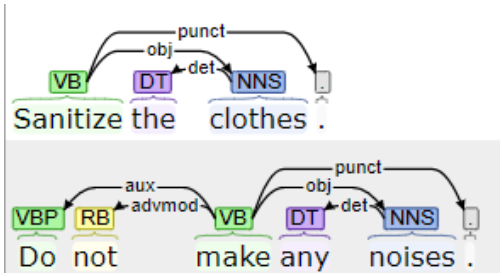
Instruction: Wash my silk blouse with cold water. Use extra light soil level.

After parsing:



Instruction: Sanitize the clothes. Do not make any noises.

After parsing:



3.1.3. Grammar

Table of POS tag abbreviations:

POS tag	Meaning	POS tag	Meaning
NN	Noun	PRP	Possessive pronoun
VB	Verb	RB	Adverb
VBP	Auxiliary Verb	IN	Preposition
DT	Determinant	CD	Cardinal number
JJ	Adjective	TO	Preposition

Terminals : NN, VB, VBP, DT, JJ, PRP, RB, IN, CD, TO

Non-Terminals : NP, VP, PP

Grammar Rules

S → VP

VP → VB

VP → VB JJ

VP → VB NP

VP → VB NP PP

VP → VBP RB VB

NP → NN

NP → DT NP

NP → PRP NP

NP → JJ NP

NP → NP VP

NP → NP PP

NP → RB JJ NN

PP → IN

PP → TO

PP → IN CD NN

PP → PP NP

PP → PP VP

Notes:

1. In our grammar, we described all compound words as one word. Some examples are shown in the following table.

Word	**Original tag**	**Tag used in our grammar**
soil level	NN, NN	NN
spin speed	NN, NN	NN
extra light	JJ, JJ	JJ

Since our system allows only 'imperative' sentences, all the input instructions must start with a verb. This is our first grammar rule [S □ VP].

The verb phrase of our grammar are as follows:

VP → VB

VP → VB JJ

VP → VB NP

VP → VB NP PP

VP → VBP RB VB

We assumed that our verb phrase can be in five different forms:

1. Only a simple verb. E.g. "Clean", "Wash", "Sanitize", "Use", "Start".
2. A verb together with an adjective. E.g. "Be/VB quiet/JJ".
3. A verb followed by a noun phrase. E.g. "Clean/VB my/PRP chiffon/NN".
4. A verb followed by a noun phrase and a prepositional phrase. E.g. "Clean/VB my/PRP chiffon/NN at/IN 9/CD am/NN".
5. A negative verb. E.g. "Do/VB not/RB leave/VB any/DT wrinkles/NNS".

The noun phrase of our grammar are as follows:

NP → NN
NP → DT NP
NP → PRP NP
NP → JJ NP
NP → NP VP
NP → NP PP
NP → RB JJ NN

Here, we assumed that our noun phrase can be in seven different forms:

1. A simple noun. E.g. “chiffon”, “clothes”.
2. A noun phrase with a determinant. E.g. “the/DT clothes/NNS”, “the/DT dirty/JJ shirt/NN”.
3. A noun phrase with a pronoun. E.g. “my/PRP clothes/NNS”, “these/PRP white/JJ shirts/NNS”.
4. A noun phrase with an adjective. E.g. “dirty/JJ shirts/NNS”, “max/JJ spin/NN speed/NN”
5. A noun phrase with a verb phrase. E.g. “clean/VB clothes/NNS using/VBG delicates/NN cycle/NN”.
6. A noun phrase with a prepositional phrase. E.g. “use/VB steam/NN to/TO clean/VB these/PRP shirts/NN”.
7. An adjective noun with an adverb. E.g. “heavily/RB dirty/JJ clothes/NNS”.

The prepositional phrase of our grammar are as follows:

PP → IN
PP → TO
PP → IN CD NN
PP → PP NP
PP → PP VP

We assumed that our preposition phase can be in five different forms:

1. Rule 1 and 2: Simple preposition. E.g. “with./IN”, “after/IN”, “to/TO”
2. Preposition for delay time with a preposition, cardinal number and a noun. E.g. “at/IN 3/CD pm/NN”, “at/IN 10/CD am/NN”.
3. Preposition with a noun phrase. E.g. “at/IN high/JJ spin/NN speed/NN”.
4. Preposition with a verb phrase. E.g. “to/TO clean/VB these/PRP shirts/NN”.

3.2. Translation

Our system proposed nine steps to translate natural language instructions into machine understandable language. Firstly, all the words in the input instruction are converted into lowercase and their spelling are checked and corrected. Secondly, the preprocessed instruction is passed through a dependency parser to get the relations between pairs of words. Thirdly, we extract two lists named ‘M’ and ‘T’ using a rule-based matching approach on relations between word pairs. ‘M’ is a list with words that describe user-defined cycles and options. ‘T’ is a list with words that includes delay (start operating time). As a fourth step, we search for the keywords of cycle names in ‘M’. We presumed it is a ‘Normal' cycle if the user did not specify a particular cycle. In the fifth stage, we extract user-defined functions for water temperature, spin speed, soil levels, and five additional options such as ‘fresh rinse’, ‘prewash’, ‘wrinkle release’, ‘control lock’ and ‘mute sound’ from ‘M'. In the sixth step, the default functions of the resulting cycle are set if there is any function which is not defined by the user. We extract the user-defined delay time from T in the seventh step. Users can set delay time in two formats which are ‘after n hours and m minutes’ or ‘at hh:mm am/pm’. If the user does not define any specific time, then we set the current time as the start operating time.

As an eight step, we check if user-defined functions and extra options are available for the cycle or not. If any of them is unavailable, then we notify users and set them to default of the specific cycle. Finally, we compile all of the information we've gathered and check with the user that it's what he or she intends. If everything verifies correctly, we give the collection of instructions to the machine. Otherwise, we'll have to ask the user for a new command.

Step 1: Convert to lowercase

First we convert all words into lowercase using python string methods .lower().

Step 2: Get Dependency Relations of words in the instructions

In step 2, a NLP python library, Stanza, was used to create a processor pipeline to process the instructions and create different attributes. The pipeline we used contains the following process.

Name	Processor class name	Requirement	Generated Annotation	Description
tokenize	Tokenize Processor	-	Segments a Document into Sentences, each containing a list of Tokens	Tokenizes the text and performs sentence segmentation.
mwt	MWT Processor	tokenize	Expands multi-word tokens (MWTs) into multiple words when they are predicted by the tokenizer. Each Token will correspond to one or more Words after tokenization and MWT expansion.	Expands multi-word tokens (MWT) predicted by the TokenizeProcessor.
pos	POS Processor	tokenize, mwt	UPOS, XPOS, and UFeats annotations are accessible through Word's properties pos, xpos, and ufeats.	Labels tokens with their universal POS (UPOS) tags, treebank-specific POS (XPOS) tags, and universal morphological features (UFeats).
lemma	Lemma Processor	tokenize, mwt, pos	Perform lemmatization on a Word using the Word.text and Word.upos values. The result can be accessed as Word.lemma.	Generates the word lemmas for all words in the Document.
depparse	Depparse Processor	tokenize, mwt, pos, lemma	Determines the syntactic head of each word in a sentence and the dependency relation between the two words that are accessible through Word's head and deprel attributes.	Provides an accurate syntactic dependency parsing analysis.

The output of the Pipeline is a Document which is composed of Sentences which is a python list, each containing dicts of Tokens with its attributes.

Example of a dict of one word from the instruction and its attributes.


```
{
  "id": 1,
  "text": "Clean",
  "lemma": "clean",
  "upos": "VERB",
  "xpos": "VB",
  "feats": "Mood=Imp|VerbForm=Fin",
  "head": 0,
  "deprel": "root",
  "misc": "start_char=0|end_char=5"
 },
```


These are the attributes that were used in extraction of M and T.

Attribute Name	**Definition**
'id'	id of the word in the current sentence. The first word of the sentence has 'id' of 1.
'text'	text of the current word
'lemma'	lemma of the current word
'xpos'	treebank-specific POS (XPOS) tag of the current word
'upos'	universal POS (UPOS) tag of the current word. (The possible UPOS tags are ADJ: adjective, ADP: adposition, ADV: adverb, AUX: auxiliary, CCONJ: coordinating conjunction, DET: determiner, INTJ: interjection, NOUN: noun, NUM: numeral, PART: particle, PRON: pronoun, PROPN: proper noun, PUNCT: punctuation, SCONJ: subordinating conjunction, SYM: symbol, VERB: verb, X: other)
'head'	'id' of the head word of the current word. (if 'id' is 0 then the head word is 'root')
'deprel'	dependency relationship between the current word and the head word

Step 3: GET ‘M’ and ‘T’

In this step, we get the value of modificators, M and Temporal modificators, T based on rules. The rules consider the POS tags and/or the type of dependency relationship between the words. If a pair of words satisfies the condition of our rules we append the lemma of the words to the M or T list.

Abbreviation

- VB+ = 'VB', 'VBP', 'VBG', 'VB'
- NN+ = NN/ NNS/ every NN
- +mod = amod/ advmod/ every mod

Create Lists ‘M’ and ‘T’ using get_MT() function

Create a list ‘M’ using the following concepts:

1. Find an 'obj' dependency relationship that has a verb as its head. A word that has a ‘compound’ relationship with the object is also taken into M. This is to detect the instruction where the user specifies the options with a verb.
2. Get word pairs that have a ‘compound’ relationship and also the word that has a relationship that contains ‘mod’ in the name with the head of the pair, for example, ‘nmod’, ‘amod’ and ‘advmod’. All ‘mod’ relationships are accepted except 'nmod:poss’ because the relationship about possession does not affect the options.
3. Get word pairs that have relationships that contain ‘mod’ in the name except 'nmod:poss’.
4. Get word pairs that have ‘aux’ relationships and both words’ POS tags are VB+.
5. Get word pairs that have ‘aux’ relationships and both words’ POS tags are VB+ and also get words with POS tag ‘RB’ that have ‘advmod’ relationship with the head verb. If there is a word with ‘obj’ relationship with the head verb then we also take it.
6. Find a word that has a relationship ‘case’ and has a POS tag ‘IN’ and its head is ‘NN’. Then find another pair of words that have a relationship ‘obl’ and the head is VB+ and also take its ‘mod’. Then we also take the ‘obj’ of the VB+ and its ‘compound’.
7. Get word pairs that have a relationship ‘cop’ and the head is ‘JJ’ and the tail is ‘VB’.

Create a list ‘T’ using the following concepts:

1. Get a set of ‘IN’ , ‘CD’ and ‘NN’ that appear in sequential order. If there are other words between them, if the order is still preserved we still append them into the ‘T’ list.
2. Get pairs of words that have the relationship ‘nummod’. This rule is for detecting the delay time.

After the ‘M’ and ‘T’ lists are extracted from the instruction, we check for any duplicates and remove them. In the next steps we will use the stems of the ‘cycle’ name, ‘function’ name, ‘selection’ and ‘extra option’ name to search for user-specified options from the word pairs in M list. A Snowball Stemmer from the nltk library was used.

Step 4: Get Cycle

The variable for the selected ‘cycle’ is ‘fin_cycle’. We first initialize ‘fin_cycle’ as None, thean the 'set_cycle’ function will search if information about the cycle is mentioned in the pairs in M list, then set the value of ‘fin_cycle’ accordingly. The python re (regular expression) library was used to search the string for a match.

1. There are some clothes types restricted to a specific cycle. If we found those clothes types in M, we will set them to the respective specific cycle. They are:
    a. delicate_stem = ['nylon', 'silk', 'wool', 'cotton', 'linen', 'chiffon', 'georgett', 'moir', 'ninon']
    b. activewear_stem = ['gym', 'jogger', 'polyest', 'sport', 'tracksuit', 'leg','yoga', 'workout']
    c. color_stem = ['dye']
2. Users can input specific cycles in any word form. For example, users can say ‘delicately, delicates, delicate’ to mention ‘delicates’ cycle. Therefore, we search for the stem of a specific cycle using ‘one_cycle_stem’ = ['normal', 'activewear', 'color', 'delic', ‘handwash', 'favorit', 'steam', 'refresh', 'jean', 'duti', 'dirty', 'sanit', 'germ', 'white', 'quick']
3. Some of our menu cycles are in two-word forms which are [['fast', 'wash'], ['whitest', 'white'], ['heavi','duty'], ['rins', 'spin'], ['rins', 'cycl'], ['spin', 'cycl']]. Hence, we also have ‘two_cycle_stem’. We find the match for word pairs of ‘two_cycle_stem’ with the pairs of M. If there is a match, we take two words as a cycle.

4. After getting a stem of a specific cycle, we transform it into the manual-defined cycle using ‘cycle_dict’.

5. If no specific cycle is defined and ‘fin_cycle’ is still None at the end of the search we set the ‘fin_cycle’ as ‘normal’.

Note:

- For ‘rinse & spin’ cycle we only search for the word ‘rinse and spin’ that comes consequently without other words in between them to avoid confusion with the ‘spin speed’ and ‘fresh rinse’. We also search for a pair in the M list that contains the word ‘no’ and ‘detergent’ then imply that as ‘rinse & spin’ cycle.

Step 5: Set User-defined Functions And Options

At first we initialized the ‘temperature’, ‘spin_speed’, ‘soil_level’ and ‘option_fin’ as None. Then we try to extract user-defined selections for each ‘function’.

Step 5.1: Get Temperature

1. We search for the stem of the function name first which may be ‘water’ or ‘temperatur’.
2. If there is a match, we search the stem of the selection value ['sanit', 'hot', 'warm', 'cold', 'tap'] in the match pair of M.
3. If there is a match, we saved it as a user-defined water temperature.

Note : To avoid confusion between ‘tap cold’ and ‘cold’ water, we only search for ‘tap’ and the temperature is set as ‘cold’ only when ‘cold’ is detected in the instruction and ‘tap’ is not detected in the instruction.

Step 5.2: Get Spin Speed

1. We search for the stem of the function name first which may be ‘spin’ or ‘speed’.
2. If there is a match, we search the stem of the selection value ['max', 'high', 'medium', 'low', 'no'] in the match pair of M.
3. If there is a match, we saved it as a user-defined spin speed.

Note : If ‘no’ is found in the same pair as ‘spin’ then the ‘spin speed’ is set as ‘no spin’

Step 5.3: Get Soil Level

1. We search for the stem of the function name first which may be ‘soil’ or ‘level’.
2. If there is a match, we search the stem of the selection value ['max', 'heavy', ‘heavi’,’'medium', 'light', 'extra'] in the match pair of M.
3. If there is a match, we saved it as a user-defined soil level.

Note : Both ‘heavy’ and ‘heavi’ are searched for and converted to ‘heavy’. In ‘steam refresh’ and ‘rinse & spin’ the ‘soil level’ is automatically set as None.

Step 5.4: Get The Extra Options

- There are five ‘extra options’ that users can set in our machine. Users can do ‘fresh rinse’, ‘prewash’, ‘wrinkle release’, ‘control lock’ and ‘mute sound’.
- We assume that the user will mention above five options only when he or she wants that option.
- Since we are using the same ‘M’ list for both defining ‘cycle’ and ‘extra options’, we can miss recognizing the ‘fresh rinse’ option and ‘rinse & spin’ cycle. Therefore, we set ‘fresh rinse’ as ‘two_option_stem’; which restricts it to have both ‘fresh’ and ‘rinse’ in the pair of ‘M’.
- We search for keywords ['wrinkl', 'control', 'lock', 'prewash', 'mute', 'sound', 'quiet', 'loud', 'nois'] in ‘M’ list.
- After getting a stem of a specific ‘extra option’, we transform it into the manual-defined ‘extra option’ using ‘option_dict’.

Step 6: Set Default Options

After extracting the user-defined ‘selection’ in step 5 if any ‘selection’ of the ‘function’ is still None, we set them to the default selection of that ‘cycle’.

- There are default settings for ‘temperature’, ‘spin speed’ and ‘soil level’ for each ‘cycle’ except ‘my favorite’ cycle.
- ‘my_favorite’ cycle may vary based on individual users. Therefore, we assumed it’s default settings were the same as the normal cycle in our system.

- We will set the default one for any function that the user does not mention. For example, if a user says ‘Wash my clothes in a ‘delicates’ cycle. The system will automatically wash the clothes using a ‘delicates’ cycle, ‘warm’ water, ‘medium’ spin speed and ‘medium’ soil level.

Step 7: Set Time

Users may define delay time in two ways: ‘after n hour and m minutes’ or ‘at hh:mm am/pm’.

Case 1: ‘After specific duration’

1. Firstly, we extract user-defined hours and minutes.
2. If there is ‘half’ in T: we marked it as after 30 minutes.
3. Then, we search for ‘hour’ or ‘hours’ in the second part of pairs of ‘T’. If there is a match, we take the first part of that pair, which is a number describing user-defined hours.
4. Then, we search for ‘minutes’ in the second part of pairs of ‘T’. If there is a match, we take the first part of that pair, which is a number describing a user-defined minute.
5. We retrieve the current hour and minute. Then, user-defined time is added to current time.
6. If it is over 24 hours, then we mark it as a to-do list for tomorrow.
7. Finally, we changed it into am/pm format.

Case 2: ‘At definite time (am/pm)’

1. Check whether the set time is prior now or not.
2. If it is, mark it as ‘tomorrow’.

Step 8: Check for Unavailable Function Or Option In Specific Cycle

We have ‘conditions’ as a list containing dicts for each cycle, each containing the possible ‘selection’ for the ‘temperature’, ‘spin speed’, ‘soil level’ and ‘extra options’. Then the ‘check_condition’ function checks if the specified ‘temperature’, ‘spin speed’, ‘soil level’ and the ‘extra options’ is available for the selected ‘cycle’, if not it will be changed to the default ‘selection’ of that ‘function’ and a note will also be displayed with the output.

Step 9: Ask for Confirmation

The final information is shown to the user to check whether it's what he or she intends. If everything verifies correctly, we give the collection of instructions to the machine. Otherwise, we'll have to ask the user for a new command.

4. Results

Our system enables users to wash using any cycle or options they desire without having to understand the basic features of the machine. Some of the unique features are highlighted with examples. The system is capable of comprehending a variety of user instructions. For instance, the device may recognize when a user mentions a particular cycle name. The user explicitly listed steam refresh in the instruction, and the device chose the ‘steam refresh' cycle in the output.

```
Enter command : Wash my dresses with steam refresh. Lock the control.
==================================================
Input :  wash my dresses with steam refresh. lock the control.
==================== Output ====================
- Cycle :  STEAM_REFRESH
- Modification/Options :
   > [HOT] Water
   > [NO SPIN] Spin Speed
   > [NONE] Soil Level
   > Control Lock
- Time :  Start at 01:10 pm
* Notes :  []
*** Please confirm the above command ***
[yes] or [no]  :  yes
==================================================
Command Accepted !
Executing the above command...
```

User-defined options for available configurations can also be accommodated by the system. For example, since the user requested tap cold water, the system selects ‘tap cold' as the water temperature. But since the user did not specify spin speed or soil level, the device chooses the options default to the ‘activewear' cycle, high spin speed, and medium soil level. Users can also choose other available options such as ‘wrinkle release’.

```
Enter command : Use tap cold water to clean my sports clothes. Do not leave any wrinkle.
==================================================
Input :  use tap cold water to clean my sports clothes. do not leave any wrinkle.
==================== Output ====================
- Cycle :  ACTIVEWEAR
- Modification/Options :
   > [TAP COLD] Water
   > [HIGH] Spin Speed
   > [MEDIUM] Soil Level
   > Wrinkle Release
- Time :  Start at 11:53 am
* Notes :  []
*** Please confirm the above command ***
[yes] or [no]  :  yes
==================================================
Command Accepted !
Executing the above command...
```

The system can also detect when the clothes need special attention during a wash. The user in the example mentioned that the fabric was a silk blouse, and we all know that silk clothing is delicate. As a result, the machine selected the ‘delicate' cycle for it.

```
Enter command : Wash my silk blouse with cold water. Use extra light soil level.
==================================================
Input :  wash my silk blouse with cold water. use extra light soil level.
==================== Output ====================
- Cycle :  DELICATES
- Modification/Options :
   > [COLD] Water
   > [MEDIUM] Spin Speed
   > [EXTRA LIGHT] Soil Level
- Time :  Start at 01:07 pm
* Notes :  []
*** Please confirm the above command ***
[yes] or [no]  :  yes
==================================================
Command Accepted !
Executing the above command...
```

Furthermore, the system has the ability to change if the user gives an incorrect instruction. In this case, the user specifies that the handwash cycle is performed with cold water. However, only ‘warm' water is available as a water temperature in the handwash cycle. As a result, the machine chooses warm water and notes it in the notes section.

```
Enter command : Handwash the shirt with cold water after 30 minutes.
==================================================
Input :  handwash the shirt with cold water after 30 minutes.
==================== Output ====================
- Cycle :  HAND_WASH
- Modification/Options :
   > [WARM] Water
   > [LOW] Spin Speed
   > [MEDIUM] Soil Level
- Time :  Start at 00:29 pm
* Notes :  ['cold water not available in hand_wash cycle, Automatically changed to warm.']
*** Please confirm the above command ***
[yes] or [no]  :  yes
==================================================
Command Accepted !
Executing the above command...
```

Moreover, the user has the option of delaying the wash cycle. As can be seen in the example, the user commanded that the device wash the clothes after 2 hours. As a result, the device set the start time to 2 hours from the time the instruction was given.

```
Enter command : Wash these jeans with tap cold water after 2 hours.
==================================================
Input :  wash these jeans with tap cold water after 2 hours.
==================== Output ====================
- Cycle :  JEANS
- Modification/Options :
   > [COLD] Water
   > [MEDIUM] Spin Speed
   > [MEDIUM] Soil Level
- Time :  Start at 03:23 pm
* Notes :  ['tap cold water not available in jeans cycle, Automatically changed to cold.']
*** Please confirm the above command ***
[yes] or [no]  :  yes
==================================================
Command Accepted !
Executing the above command...
```

Our systems still have some drawbacks. First and foremost, the device is tailored to the Electrolux washing machine. So, in order to apply this method to other washing machines, we'll need to look at their systems. Then when the system has a choice between two alternatives, it selects the first alternative. Both ‘activewear' and ‘heavy duty' cycles may be choices for the example given below, as the user listed gym clothes and that they are very dirty. As a result, the device chooses 'heavy duty' as the cycle.

```
Enter command : wash my dirty jeans clothes. use max spin speed.
================================================
Input :  wash my dirty jeans clothes. use max spin speed.
==================== Output ====================
- Cycle :  HEAVY_DUTY
- Modification/Options :
   > [WARM] Water
   > [MAX] Spin Speed
   > [MEDIUM] Soil Level
- Time :  Start at 00:15 pm
* Notes :  []
*** Please confirm the above command ***
[yes] or [no]  :  yes
================================================
Command Accepted !
Executing the above command...
```

Finally, in certain circumstances, the machine is unable to choose the ‘color' cycle. If the user explicitly specifies the color name instead of mentioning that it is colorful clothing, the device does not select the ‘color' cycle.

```
Enter command : wash my blue dress with light soil level. Mute the sound.
================================================
Input :  wash my blue dress with light soil level. mute the sound.
==================== Output ====================
- Cycle :  NORMAL
- Modification/Options :
   > [WARM] Water
   > [MAX] Spin Speed
   > [LIGHT] Soil Level
   > Mute Sound
- Time :  Start at 00:21 pm
* Notes :  []
*** Please confirm the above command ***
[yes] or [no]  :  yes
================================================
Command Accepted !
Executing the above command...
```

5. Conclusion

In this paper, we presented a software architecture to allow users to interact with their smart washing machine. The proposed system is capable of interpreting a command given in an imperative command structure and extracting the meaning of all of its meanings. Furthermore, our Natural Language Processor operates independently of the text message because it analyzes the grammar structure of the sentence in detail using a dependency parser rather than comparing it to prepared texts. We can parse the sentence and understand its context (intentions and entities). Furthermore, our system can handle a flow of intentions with conditionals. However, as natural language has such a large number of grammatical structures, our grammar's structural format is relatively small.

References

1. A. Xu, Z. Liu, Y. Guo, V Sinha, R. Akkiraju, “A New Chatbot for Customer Service on Social Media”, In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, pp. 3506–3510, 2017.
2. C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, “Learning to parse natural language commands to a robot control system”, Experimental Robotics, Springer International Publishing, pages 403–415, 2013.
3. D. K. Misra, J. Sung, K. Lee, and A. Saxena, “Tell me Dave: Context sensitive grounding of natural language to manipulation instructions”, In *Proceedings of Robotics: Science and Systems (RSS)*, Berkeley, USA, July 2014.
4. D. Seo and J. Lee, “Web 2.0 and five years since: How the combination of technological and organizational initiatives influences an organization’s long-term Web_2.0 performance”, In *Telematics and Informatics*, vol.33, no.1, pp.232-246, 2016.
5. G. M. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, H. Zender, I. Kruijff-Korbayová, N. Hawes, “Situated dialogue processing for human-robot interaction”, In *Cognitive Systems*, 2010.
6. J. Thomason, S. Zhang, R. Mooney and P. Stone “Learning to Interpret Natural Language Commands through Human-Robot Dialog”, In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2015),* Buenos Aires, Argentina, July 2015.
7. L. She, S. Yang, Y. Cheng, Y. Jia, J. Chai, and N. Xi, “Back to the blocks world: Learning new actions through situated human robot dialogue”, In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 89–97, June 2014.
8. P. Jensfelt, G. M. Kruijff, H. Zender and H. I. Christensen, “Situated dialogue and spatial organization: What, where and why?”, In *International Journal of Advanced Robotic Systems*, 2007.
9. S. A. Abdul-Kader and J. Woods, “Survey on chatbot design techniques in speech conversation systems,” In *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, vol.6, no.7, 2015.