Interactors

1. Single List Selection

Read the list of items and be ready to accept a response (yes, yeah, that one) after each item. Only one item from the list can be selected.

Input: String initial prompt, String comma-separated list of options, id of output control
Output: String representing user choice
Action:

1. Read the initial prompt
2. Read the list of options, allowing user to interrupt
3. If user interrupts with "yes', "yeah", or "that one", interrupt playback and return the name of the most recently-read choice
4. If user interrupts with another response, continue reading
5. If no response is given, return null

Characteristics: more efficient, improves prompting and turn around time of listening, no need to remember the whole list: easier for the user.

e.g. Prompt: "Would you like to buy apples, oranges, bananas?" The user: "That one" after hearing oranges".

2. Multiple List Selection (using speech)

Read the list of items and be ready to accept a response (yes, yeah, that one) after each item. After user selects an item, prompt to see if he/she wants to select other item. If yes, then read the rest of the list listening for the response. Repeat after each selected item until the end of the list is reached. One or more items can be selected.

Input: String initial prompt, String comma-separated list of options, id of output control, string intermediate prompt, optional String continuation prompt
Output: String representing user choices, separated by commas
Action:

1. Read the initial prompt
2. Read the list of options, allowing user to interrupt
3. If user interrupts with "yes', "yeah", or "that one", interrupt playback and return the name of the most recently-read choice
    i. Add the selection to the output string
    ii. Prompt user with "you selected", followed by the text of their choice
    iii. If additional choices remain in the list, do the following:
        1. If a continuation prompt was provided, read the continuation prompt
        2. Otherwise, prompt with "Would you like to make an additional selection?"

3. If the user answers “yes’ or “yeah”, read the intermediate prompt and go back to step 2, starting from the next item in the list
4. Return the output string

4. If user interrupts with another response, continue reading
5. If no response is given, return null

Characteristics: more efficient, improves prompting and turn around time of listening, the list is read in pieces: easier for the user to remember.

*e.*g. Prompt: “Would you like to buy apples, ..?” User says “yes” after hearing apples. Computer response: “You selected apples. Would you like to buy something else?” User says “yes”. Prompt: “oranges, bananas”. User says “yeah” after hearing bananas. Computer response: “You selected bananas”. (the end of the list is reached)

3. Multiple List Selection (using DTMF)

Read the list of items and be ready to accept a response (pushed button 1) after each item. After user selects an item, ask him/her to push 2 to select other item or to push 3 to finish selection. If pushed 2, then read the rest of the list listening for the response. Repeat after each selected item until user pushes 3 or until the end of the list is reached. One or more items can be selected.

Input: String initial prompt, String comma-separated list of options, id of output control, string intermediate prompt, optional String continuation prompt
Output: String representing user choices, separated by commas
Action:

1. Read the initial prompt
2. Read the list of options, allowing user to interrupt
3. If user interrupts with “1”, interrupt playback and return the name of the most recently-read choice
    i. Add the selection to the output string
    ii. Prompt user with “you selected”, followed by the text of their choice
    iii. If additional choices remain in the list, do the following:
        1. If a continuation prompt was provided, read the continuation prompt
        2. Otherwise, prompt with “Press 2 to make additional selections, press 3 if you are finished.””
        3. If the user answers with a “2”, read the intermediate prompt and go back to step 2, starting from the next item in the list
        4. Return the output string
4. If user interrupts with another response, continue reading
5. If no response is given, return null

Characteristics: more efficient, improves prompting and turn around time of listening, the list is read in pieces: easier for the user to remember, more accurate understanding of responses, added complexity: have to explain what 1, 2, or 3 means.

e.g. Prompt: “Would you like to buy apples, ..?” User pushes 1 after hearing apples. Computer response: “You selected apples. Press 2 if you would like to buy something else, or press 3 to finish” User pushes 2. Prompt: “oranges, bananas”. User pushes 1 after hearing bananas. Computer response: “You selected bananas”. (The end of the list is reached)

4. Multiple List Selection (using speech) With Deselecting

Read the list of items and be ready to accept a response (yes, yeah, that one) after each item. After user selects an item, prompt to see if he/she wants to select other item. If yes, then read the rest of the list listening for the response. Repeat after each selected item until the end of the list is reached. When user is done selecting, read the list of selected items back to him/her, and instruct the user to say “no” after the item they would like to deselect. One or more items can be selected or deselected.

Input: String initial prompt, String comma-separated list of options, id of output control, string intermediate prompt, optional String continuation prompt, optional string verification prompt
Output: String representing user choices, separated by commas
Action:

1. Read the initial prompt
2. Read the list of options, allowing user to interrupt
3. If user interrupts with “yes’, “yeah”, or “that one”, interrupt playback and return the name of the most recently-read choice
    i. Add the selection to the output list
    ii. Prompt user with “you selected”, followed by the text of their choice
    iii. If additional choices remain in the list, do the following:
        1. If a continuation prompt was provided, read the continuation prompt
        2. Otherwise, prompt with “Would you like to make an additional selection?”
        3. If the user answers “yes’ or “yeah”, read the intermediate prompt and go back to step 2, starting from the next item in the list
        4. Go to step 6
4. If user interrupts with another response, continue reading
5. If no response is given, return null
6. If a verification prompt was provided, read the verification prompt, otherwise read “You made the following selections – you may remove any selection by saying ‘no’ after the selection is read.”
7. Read back each item in the output list, allowing the user to interrupt
8. If the user interrupts with “no”, continue reading from next item
9. Otherwise add the item to the output list

Characteristics: more efficient, improves prompting and turn around time of listening, the list is read in pieces: easier for the user to remember, allows the user to confirm the selection and get rid of items they did not mean to select.

*e.*g. Prompt: "Would you like to buy apples, ..?" User says "yes" after hearing apples. Computer response: "You selected apples. Would you like to buy something else?" User says "yes". Prompt: "oranges, bananas". User says "yeah" after hearing bananas. Computer response: "You selected bananas". (The end of the list is reached) Prompt: "Here are the items you selected. Say no after an item you don't want to keep. Apples, Oranges". User says no after bananas. (Bananas are taken off the list)

5. Double Yes/No (using speech)

When asked yes/no question, the user says something else (yeah, may be, neh …) Will confirm unclear response by asking "Did you say Yes/No?" Will be used on error (user says something but yes/no) or on low confidence level (user says something close to yes/no). Repeat the confirmation only once.

Input: String initial prompt, id of output control, optional String 'yes' prompt, optional string 'no' prompt
Output: String representing user choice
Action:

1. Read the initial prompt and wait for a response
2. If "yes" or "yeah" is chosen with low confidence
    i. Read the "yes" prompt if one was provided
    ii. Otherwise ask "was that a 'yes'?"
    iii. Wait for response
    iv. If no, return "no"
    v. If "yes", return "yes"
    vi. If error, go to step 1
    vii. If no response, return "yes"
3. If "no" is chosen with low confidence
    i. Read the "no" prompt if one was provided
    ii. Otherwise ask "was that a 'no'?"
    iii. Wait for response
    iv. If no, return "yes"
    v. If "yes", return "no"
    vi. If error, go to step 1
    vii. If no response, return "no"
4. If "yes" or "yeah" is chosen with high confidence, return "yes"
5. If "no" is chosen with high confidence, return "no"
6. If the response does not match the grammar, return null
7. If no response is given, return null

Note: ask user 10 yes/no questions and see how many times an error or low confidence level occurs. This will give an idea if this interactor is actually useful.

e.g. Prompt: "Would you like a single room?" User says "Ya". Computer response: "Did you say yes?" User says "Yes". (The answer to the original question was yes) OR In response to "did you say yes" user says "No". (The answer to the original question was no)

6. Double Yes/No (using DTMF)

When asked yes/no question, the user says something else (yeah, may be, neh …) Will confirm unclear response by saying "If you said yes, press 1, if you said no, press 2". Will be used on error (user says something but yes/no) or on low confidence level (user says something close to yes/no). Repeat the confirmation only once.

Note: ask user 10 yes/no questions and see how many times an error or low confidence level occurs. This will give an idea if this interactor is actually useful.

Characteristics: will be clearer than Double Yes/No using Speech, but does it make sense to use?

Input: String initial prompt, id of output control, optional String 'yes' prompt, optional string 'no' prompt
Output: String representing user choice
Action:

1. Read the initial prompt and wait for a response
2. If "yes" is chosen with low confidence
    i. Read the "yes" prompt if one was provided
    ii. Otherwise ask "press one if your answer was 'yes', press 2 if your answer was 'no'"
    iii. Wait for response
    iv. If 1, return "yes"
    v. If 2, return "no"
    vi. If no response, return "yes"
3. If "no" is chosen with low confidence
    i. Read the "no" prompt if one was provided
    ii. Otherwise ask "press one if your answer was 'yes', press 2 if your answer was 'no'"
    iii. Wait for response
    iv. If 1, return "yes"
    v. If 2, return "no"
    vi. If no response, return "no"
4. if "yes" or "yeah" is chosen with high confidence, rreturn "yes."
5. if "no" is chosen with high confidence, return "no."
6. If error, return null
7. If no response is given, return null

e.g. Prompt: "Would you like a single room?" User says "Ya". Computer response: "If you said yes, press 1, if you said no, press 2?" User presses 1. (The answer to the

original question was yes) OR User presses 2. (The answer to the original question was no)

7. N-Best List

Confirm an answer to a "textual" question using a proximity threshold in the n-best list. After user's response, get the list of words that match that response the most, and prompt the user again with that list of words: "Did you say item1, item2 or item3?"

Input: String initial prompt, String comma-separated list of options, id of output control, optional real number threshold
Output: String representing user choice
Action:

1. Read the initial prompt, wait for user response
2. Examine returned n-best list
3. If one or more options is within threshold of top choice
    i. Read each item, allowing user to interrupt
    ii. If user interrupts with "yes" or "yeah", return the text of the last value read
    iii. If user interrupts with something else, continue reading from the list
    iv. If user does not respond, return the top choice
4. Otherwise, return the selection made

e.g. "What city would you like to visit?" User says "Long View". Computer response: "Did you say Long View, Border Butte, or Fond Duke?" User says yes after hearing Long View. (The city user wanted to visit was Long View)

8. Re-ask Question in Different Manner

Garbage rejection triggers second prompt: re-ask the original question in different manner.

Input: String initial prompt, id of output control, string clarification prompt, optional string list of choices
Output: String representing user choice
Action:

1. Read the initial prompt, wait for user response
2. If response is valid, return it
3. If response is an error, read the verification prompt and wait for a response
    i. If response is valid return it
    ii. If no response or error, return null
4. If no response, read the verification prompt and wait for a response
    i. If response is valid return it
    ii. If no response or error, return null

Characteristics: the alternate question will possibly prompt for a clearer answer.

e.g. Prompt: “What is your size?” User does not respond. Prompt: “What size shoes do you wear?”

9. Spoken Help Prompt

Garbage rejection or word “HELP” triggers spoken help prompt.

Input: String initial prompt, id of output control, string help prompt, optional string list of choices
Output: String representing user choice
Action:

1. Read the initial prompt, wait for user response
2. If user responds with “help”, read the help prompt and wait for a response
    i. If response is valid return it
    ii. If no response or error, return null
3. If response is valid, return it
4. If response is an error, read the help prompt and wait for a response
    i. If response is valid return it
    ii. If no response or error, return null
5. If no response, read the help prompt and wait for a response
    i. If response is valid return it
    ii. If no response or error, return null

e.g. Prompt: “How many nights would you like to stay?” User: ”five”.
(nothing happens for a while) Computer: “your choices are: 4 nights, 6 nights, or 7 nights”. User: “Five nights”.
OR Prompt: “How many nights would you like to stay?” User: “Help”. Computer: “your choices are: 4 nights, 5 nights, or 7 nights”.

10. Different Prompts for Different Responses (n-ary prompt)

If user does not respond, prompt in a certain way. If user’s response does not match the grammar, prompt in a different way.

Input: String initial prompt, id of output control, string help prompt, string “no response” prompt, string “invalid response” prompt, optional string list of choices
Output: String representing user choice
Action:

1. Read the initial prompt, wait for user response
2. If user responds with “help”, read the help prompt and wait for a response
    i. If response is valid return it
    ii. If no response or error, return null
3. If response is valid, return it
4. If response is an error, read the “invalid response” prompt and wait for a response
    i. If response is valid return it

ii. If no response or error, return null

5. If no response, read the "no response" prompt and wait for a response
    i. If response is valid return it
    ii. If no response or error, return null

11. Whisper prompt

Prompt the user with sample answer at a lower volume.

Input: String initial prompt, id of output control, string whisper prompt, optional string list of choices
Output: String representing user choice
Action:

1. Read the initial prompt
2. Lower volume and read whisper prompt
3. Wait for response
4. If response is valid, return it
5. If response is in error, return null

e.g. Prompt: "What city would you like to visit". Prompt at lower volume: "Portland Oregon, Portland Maine, or Seattle Washington".

12. Fast-Forward or Rewind using voice commands

When prompt is a long sorted (alphabetized) list, the user can fast-forward or rewind to find the item they want to select. There will be a set number of items to fast-forward or rewind. User says yes, yeah or that one after hearing an item they want to select.

Input: String initial prompt, string list of choices, id of output control, optional integer advance interval
Output: String representing user choice
Action:

1. Read each item from the list of choices, allowing interrupts
2. If the user interrupts with "forward", "fast forward", or "next", jump forward the specified number of steps and resume reading
3. If the user interruptes with "rewind", or "back", or "reverse", jump backward the specified number of steps and resume reading
4. If user responds with ""yes" or "yeah", return the test of the last item read
5. If response is an error, continue reading
6. If no response, return null

e.g. Prompt: "Alabama, Atlanta…" User: "Fast-Forward", Prompt: " California, …". User selects an item. Or the same scenario, but user says "rewind".

13. "Dictionary" Lookup

While the dictionary is being read to the user, user can fast-forward or rewind to get to the desired word. User says yes, yeah, or that one after hearing an item they want to select.

Input: URL of text source, id of output control, string initial prompt
Output: string representing user choice
Action:

1. Read the initial prompt
2. Read each item in the list, allowing interrupts
3. If the user responds with a letter, jump to the first item with that letter in the list and resume reading
4. If the user interrupts with "forward", "fast forward", or "next", jump forward the predefined number of steps and resume reading – adjust predefined steps downward
5. If the user interrupts with "rewind", or "back", or "reverse", jump backward the specified number of steps and resume reading – adjust predefined steps downward
6. If user responds with ""yes" or "yeah", return the text of the last item read
7. If response is an error, continue reading
8. If no response, return null

Characteristics – allows the user to zero in on a choice by choosing the first letter – shrinks the "forward' and "back" intervals with each successive use

14. Multiple Date Field Selection

Allow user to speak a three-part date with month and day in any order. Prompt for ambiguity check.

Input: string initial prompt, id of month output control, id of day output control, id of year output control,
Output: strings representing the date
Action:

1. Read the initial prompt
2. Wait for three numeric responses – number1, number2, and year
3. if year is not given in 4-digit format, re-ask the question
4. if number1 or number 2 > 31 or if number1 and number2 > 12, re-ask the question
5. if a nomatch occurs, return null
6. If 32 > number1 > 12 and number2 < 13 return the date, with number1 as day and number2 as month
7. If number1 < 13 and 32 > number2 > 12 return the date, with number1 as month and number 2 as day
8. Otherwise prompt user with "did you mean", followed by an interpretation of the date with number1 interpreted as month, wait for a user response
9. If user answers "yes" return the date with number1 interpreted as month

10. If user answers “no,” prompt user with “did you mean” followed by the date with number1 interpreted as day, wait for user response
11. if user answers “yes”, return the date with number1 interpreted as day
12. Otherwise, return null

e.g. Prompt: “What date would you like to arrive?”. User: “eleven, twelve, 2004”. Prompt: “Did you mean December 11th 2004?” User says yes and interactor returns 12/11/2004.

15. Confirmation and Correction Dialog after all selections.

Allow user to select all items or to answer all questions, then read back their selection and allow making changes using interactor #4.

Input: string initial prompt, string structured list of control ids to check, with textual name,
Output: none
Action:

1. Read the initial prompt – it should include something like “say ‘no’ after any incorrect response”
2. Say “you entered” then the value for each control in the list, followed by “for” and the name of the control, allowing interrupts
3. If the user interrupts with “no”
    i. If the field is textual, run a basic interactor “Please enter your choice for *field-name*”
    ii. If the field is a single selection or radio button, run the list-selection dialog
    iii. If the field is a multiple selection, run the multiple-list-selection dialog
    iv. If the field is a checkbox, run the ‘double yes/no” dialog
    v. Go to step 2, starting from the next item in the list
4. If the user interrupts with something else, continue at step 2 with the next item

16. Volume Adjuster

Prompt the user: “Say up to turn the volume up, say down to turn the volume down” and perform an appropriate action. Sets the volume for all subsequent controls used on the page

Input: string initial prompt
Output: none
Action:

1. Begin reading the initial prompt – it should start with something like “say ‘up’, ‘more’, or ‘higher’ to increase the volume, or ‘down’, ‘lower’, or ‘less’ to decrease the volume, say ‘ok’, ‘good’, ‘stop’, ‘done’, or ‘finished’ when the

volume is OK", followed by 10-20 seconds or so of text, allow the user to interrupt
2. If the user interrupts with 'up', 'more', or 'higher', increase the volume
3. If the user interrupts with 'down', 'lower', or 'less', decrease the volume
4. If the user interrupts with 'ok', 'good', 'stop', 'done', or 'finished', return
5. Continue reading the prompt from the point we left off

17. Timeout

If user does not respond for some time, prompt: "Do you need more time?" If user says yes, wait for response. If no, finish the interactor. Will have an ultimate timeout after the user was prompted a certain number of times.

Input: string initial prompt, string time prompt, optional list of valid responses
Output: string representing user choice
Action:
1. Read the initial prompt
2. Wait for a response
3. If invalid response, return null
4. If good response return response
5. If no response, read the time prompt and listen for a response
6. if user responds with "yes", go back to step 2
7. Otherwise return null