## Using the speechAPI

All voice recognition is processed using an online API from SpeechAPI.com.

SpeechAPI [ref] provides a flash object, *SpeechAPI.swf*, which can be embedded onto an HTML page using a *SWFObject.* This flash object is then used to stream audio between the client and server. When the server sends a callback back to the client, the *SPeechAPI.js* code interfaces with the flash object to allow processing to be conducted on the result call-back.

Once the flash has been successfully loaded, the *onLoaded (see Figure XXX)* method is called. Importantly, when this method is called the server needs to receive the vocabulary it needs to enable recognition for a particular web page. So depending on what the flow of the program has been set for (numerical voice referencing or spoken link names), the vocabulary is either set to include certain numbers (as required by a particular page) or names (as indicated by various link ids).

When the flash component receives the command to start recognition it starts streaming audio to the server (see section xxxx). The server will try and match in given input to one of the specified vocabularies. A string result is then returned indicating a closest match result or the string, “recognition error” if no result was found.

The API also lets you set it up so that it recognises alternative words for the same thing (so one could potentially say “go up” or “scroll up” etc.). This would give the application a more natural interface. Beyond this, the API also lets you construct grammar segments in such a way that parts of a sentence are not necessarily necessary. If thus functionality was used, links could be followed in this way instead using highlighting. However, the scope was limited to highlighting at this stage so that a basic comparison between numerical referencing and spoken link name referencing could be conducted.

## Flow Control

### Controlling flow and notifications

The main processing of results from the server (including setting up connections) is conducted in *speechprocessor.js*. Since two primary streams of tests were undertaken, two main courses of the program are determined by the state of the Boolean variable, *numericalReferenced*. If this variable is true, the program will divert to a stream which performs the required functions for numerical functions. Otherwise, the code diverts on a course that is implicitly, one of spoken name referencing.

Another Boolean variable which helps determine the course of the code is the variable, *confrimationMode*. This variable is set to true every time a result is returned from the server indicating that a link should be followed or the result is interpreted as a certain command that requires confirmation. The process introduced with the confirmation mode can be seen in section XXX.

### Confirmation mode

Confirmation was a feature only introduced at Iteration 3 of the program. It was introduced to compensate for misinterpreted user commands. From the results of the earlier iterations it was clearly evident that results would often be returned as something they are not. If a result is interpreted by the speech engine as being significant in nature, that is it will result in some form of navigation, a confirmation is required of the user (so for commands “Home”, “Backwards”,””Forwards” and all link selection).Although users preferred not to have to perform a confirmation step (See Section XXX), it is at this stage necessary.

## Implementation details

### Initialising and loading

The script will first call the *initialise* function which will set up the flash, establish a connection to the server and embed the flash. From here the program will wait until the *onLoaded* call-back is called by the provided SpeechAPI.js code. The processes involved in this call-back are illustrated in Figure XXX. In here, the vocabulary is calculated using *determineTotalVocabulary* (Figure XXX) and then *determineSpecificVocab* (Figure XXX). *DetermineSpecificVocab* then calls specific functions : d*etermineSpecificConfirmationString*, determineSpecificLinkNamesString and determineSpecificNumericalString depending on whether or not confirmation mode is enabled (through the confirmation mode variable, see section XXX) or if numerical referencing is being used (See section XXX). These functions will separate a string which contains all the commands (“home” etc.) strings and all the link referencing vocabularies (“one”, “two” etc. or “dog”, “cat” etc.) from a single string which was constructed to contain both the commands and link vocabularies (Please note: the vocabulary process for all types of referencing revolved around these sort of concacentated strings because this system was used initially on numerical referencing. Having all the vacab placed in one string meant that If you wanted to add additional commands or numbers you could all do that through a single string. The other streams (namely spoken link names and confimaion mode) also revolved around this sort of concatenated string so that all the processing for all modes was similar in logic. This makes it easier to follow each flow of vocabulary processing because all undergo the same sort of processing.)

The flash component then has it’s designated vocabulary loaded onto it (which in turn loads on the server).

### Assign referable names to links

For numerical referencing, the neededNumbers array (see section XXX) stores a list of all the needed numbers. These numbers can the be aligned to the link URLs as mentioned in section XXX. To extract the actual words for spoken link names, the function *assignLinkWords* is used to extract all the link id’s (which have previously been assigned the appropriate names) which are then assigned to the array. This array’s (*neededNamesArray)* information is later used in the same way as the *neededNumbersArray*  to search for the correct index of a returned result and consequently navigate to a link (see section XXX).

### Processing the result

When the *onResult* call-back is called, the *processResult* function is called (See Figure XXX). This function performs a series of checks that determine how to treat the result. The function hieratically performs the indicated checks. The functions *tryProcessConfirmatio*n and *tryProcessResyltAsCommand* simply conduct tests to check what type of confirmation or commands have been issued and perform the appropriate actions if a command is found. For *tryProcessResultAsLinkName* and *tryProcessResultAsNumberedLink* arrays containing all the words derived from the ids and all the numbers (*neededNumbersArray*  and *neededNameArray)* are traversed to try and find the result. If a result is found, the array index is recorded. Every link URL is then assigned to an array so that each one aligns with its corresponding link name or number in the *neededNumersArray* or *neededNameArray*. The index from the traversal is then used to access and extract the appropriate URL from the links array. Once the right URL is had, the link is followed.

Appendix:

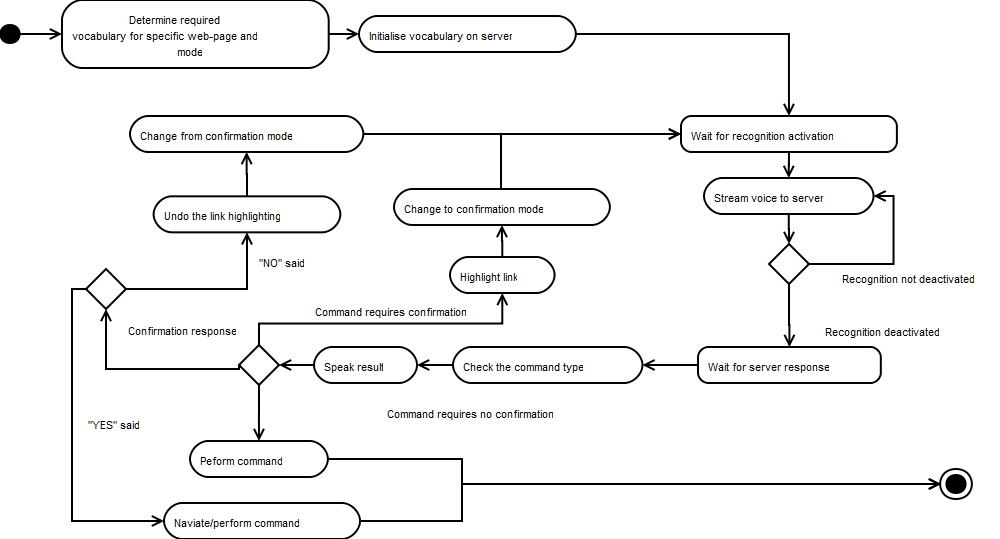


Figure A: Total overview of voice navigation

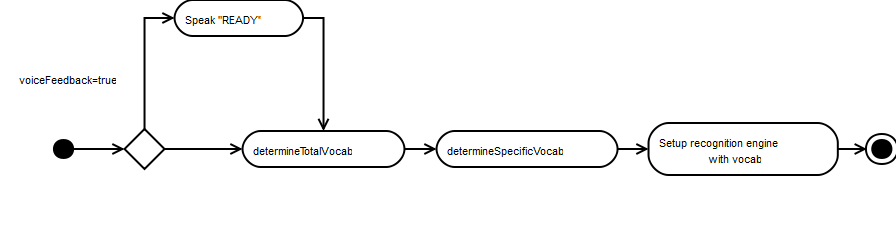


Figure : OnLoaded Overview

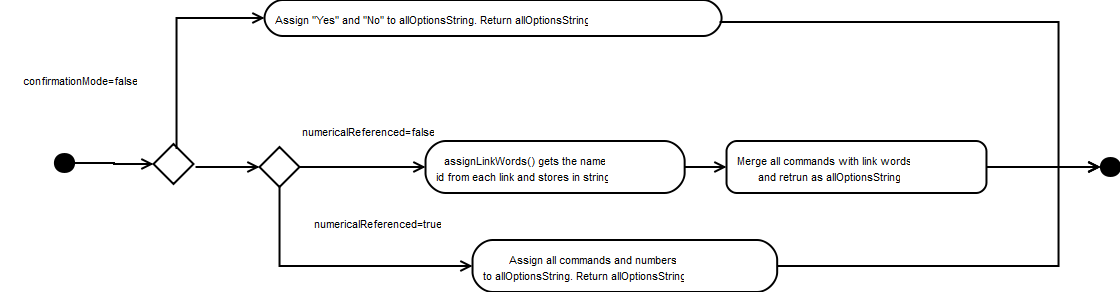


Figure : Determine total vocab overview

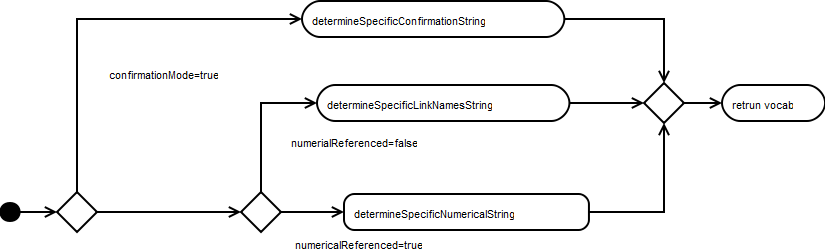


Figure :Determine specific vocab

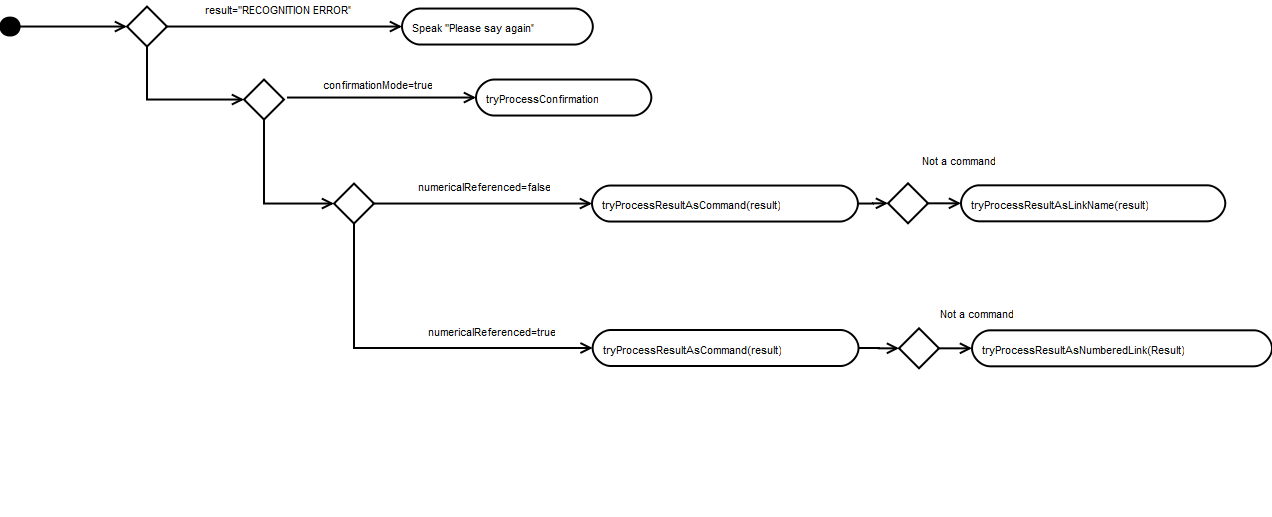


Figure : Result processing