ACV
Course Work - Implementation of an AI Component for the Letter Problem

Vicente Bosch Campos †
viboscam@posgrado.upv.es

May 3, 2011

Contents

1 Introduction **2**
1.1 Context 2
1.2 Scope 2

2 Development **2**
2.1 Program Structure 2
2.2 Conclusions 3

List of Figures

1 Introduction

1.1 Context

As part of the ACV course project we will implement an AI component in order to obtain the longest valid word that can be obtained from a set of characters.

The rules are as follows:

- The users select randomly 9 characters from the vowel or consonants groups.
- The target is to obtain the longest valid word from the Spanish language with the given letters:
  - Letters must only be used once.
  - Plurals of words are not valid
  - Personal forms of verbs are not valid.
  - Feminine forms of words are valid.

1.2 Scope

In order to implement this AI component to resolve the number problem we will develop components in order to cover the following user cases:

- Evaluate a solution provided by a user against a target value.
- Obtain a best solution through running the resolver.

2 Development

For our implementation of the sequence number problem we have implemented the algorithm in Ruby (v1.9.2) and the following libraries where used:

Awesome Print (0.3.1): Used to perform pre-formatted human readable print outs of the objects.

Timeout: Standard library object was used in order to enforce the time constraint on the execution of the genetic algorithm.

2.1 Program Structure

Next we will detail the main software components and characteristics for the sequence number resolver. The implementation can be divided in the following main software components:

Dictionary: Coded in *./lib/dictionary.rb* is the main class for this AI component. The class loads an specific word list of a language and constructs an index of the words. The dictionary class presents the following methods:

- initialize - Constructs a new object by loading the words from a word list file
- load_word_file - Is called from the initialize method and is the specific method in charge of loading the words into the hash structure. The method calculates the hash key for the specific word and adds the word to the Array of words that correspond to that hash key.
- longest_valid_words - Is the main resolver method, given a set of letters represented as a char array it studies all combinations of letters considering combinations with decreasing usage of the letters given (first it will consider words that it can build with the 9 letters given, then with all possible combinations considering groups formed by 8 letters etc) once it finds a valid word it finalizes and returns to the caller the set of valid words
- chars_to_valid_words - Returns the valid words that can be composed with a given set of letters
- chars_contain_valid_words? - Returns true if there are valid words that can be composed with a given set of letters otherwise it returns false

- word_exists? - Is used to evaluate the answers given by the user. Returns true if the word exists in the dictionary otherwise it returns false
- add_word_to_dictionary - Adds a word to the hash dictionary
- line_to_word - Converts a line in a file to a word ready for insertion
- array_to_index and word_to_index - this methods are the key to the fast resolution of this problem. Each word is turned with these methods into a key composed by the letters the word is composed of in ascending order. By using this key finding out if a word is valid of if there are any words that can be composed with a set of letters is trivial and can be resolved in constant time
- save - Allows the dictionary objet to be zipped and saved into a file for future use
- self.load - Restores a dictionary object form a previously saved instance using the save methods. By loading the dictionary with this method the load time for the game server improves significantly as the word list server is not processed each time.

2.2 Conclusions

We have developed an easy and reusable component for the resolution of the Letters problem.