MOVIE RATER

ARTIFICIAL INTELLIGENCE PROJECT

Ahmet Zorer | CMPE 480 | December 18, 2016

INTRODUCTION

In this project, the concept of artificial intelligence is exemplified by a movie review rater system. It is assumed that a given review consists from the words in the previous reviews. Basically, the program executes by getting the weighted score of each word in the review and calculates the overall score of the review.

To solve this problem, for each word in the given review, the program finds all the occurrences of the word and the score of the review in the previous reviews. Then, it calculates weighted average of the scores by multiplying the score of the reviews that includes this word. By this way, we have a learned score for each word in given review.

Though we have scores of the words, the importance of the words in the review differs. Adjectives, for example, has generally much more effect on the overall score of the review. Therefore, we need to keep the category of the word and each word has effect proportional to its category. Eventually, the program gives reasonable scores to the given reviews.

PROG RAM INTERFACE

There are 4 files in the program. *Corpus.pl* file contains reviews, *dictionary.pl* file holds the categories of the words (i.e. “huge” is an adjective and “it” is a pronoun). There are 22 reviews and 130 defined unique words in those files for now. Weights of the categories are placed in *category.pl* file and the main prolog program is in *movieRater.pl* file.

To execute the program, there are a few options. One of them is consulting the program in SWI prolog application and executing *calculate_scores(NewReview, S)* or *calculate_scores2(NewReview, S)* predicates depending on the purpose of execution. The first method divides the weighted sum of words to the number of words. However, it is more proper to divide the weighted sum of words to the sum of weights of categories of that words which is the algorithm of second method.

Another way of running the program is directly running it on sublime text editor if prolog syntax highlighting package is installed. This time directly *main* method is called. This time, the predicates defined above should be called in the main predicate.

PROGRAM EXECUTION

A review that has only words that are in the reviews in the corpus file is given as input and the calculated score of the review is shown as output. Since the aim of the system is straightforward, this is the only manner of user interaction through the project.

INPUT AND OUTPUT

The input is given as the first parameter of the *calculate_scores* predicate and the words in it are separated by 1 space as normal. The program automatically removes any commas and dots. The words of the review is lowercased for the ease of the use as well.

PROGRAM STRUCTURE

In this section, technical details of the program are discussed. Since the program is composed of predicates, the definition of the predicates will be given in this section.

REVIEW (REVIEW, SCORE)

This predicate holds the previously written reviews and their corresponding scores in *corpus.pl* file.

WORD_CATEGORY(WORD)

This predicate returns true if the given word is in the given category (i.e. *adjective("funny")* returns true). This predicate is in *dictionary.pl* file.

DEFINED (WORD)

This predicate returns true if the given word is in the given any category. This predicate is used for checking if the word is added to dictionary file. This predicate is in *dictionary.pl* file.

WEIGHT (WORD, WEIGHT)

This predicate returns the weight of the word with respect to its category. This predicate is in *category.pl* file. If the type of word is unknown the given weight is 0.

MAIN

This predicate is called automatically when building is finished on sublime text editor. Therefore, I called *calculate_scores* method with given new review for use prolog in sublime.

CALCULATE_SCORES (+NEWREVIEW, -S).

Calculates the score of the given review by using reviews on the carpus with the formula given in the project description. This predicate under cases the given review, splits it to words and gets weighted sum of the word scores. Finally, it divides this sum to the number of words to find the expected score of the review as in the project description.

CALCULATE_SCORES2 (+NEWREVIEW, -S).

Like the previous predicate but this time the weighted sum is divided to the sum of weights of words.

SUMSCORES (WORDS, TOTALWEIGHT, SCORE)

Given the list of words, returns the sum of weights of corresponding word's category and overall score of the word list.

CALCULATE_SCORE_FROM_WEIGHTED_LIST (+WEIGHTEDLIST, -SCORE).

Given the list of weighted scores and their occurrences, returns the overall score of a word.

CALCULATE_TOTAL_SCORE_FROM_WEIGHTED_LIST (+WEIGHTEDLIST, -TOTALSCORE, -COUNT).

Given the list of weighted scores and their occurrences, returns the sum of scores and occurrences of a word.

COUNT (+ELEM, +LIST, -NUMOFOCCURENCE).

Counts the number of occurrences of X in the given list.

GET_REVIEW (-WORDS, -SCORE)

Splits reviews in corpus file as word list and returns it and the score of the corresponding review.

NOT_DEFINED (-WORDS).

Returns a list of all different words on reviews that isn't put on any category, used for producing dictionary file.

UNIQ (-UNIQUES).

Returns a list of all different words on reviews, used for producing dictionary file.

SAMPLES

Some of the sample executions are given below;

calculate_scores2("The film is a super awesome movie.", Solution). Solution = 7.703125

calculate_scores2("Terrible film, totally crap.", Solution). Solution = 1.6296296296

calculate_scores2("I personally amazed with the movie. It was really fantastic.", Solution). Solution = 7.805215419501133.

IMPROVEMENTS AND EXTENSIONS

As an improvement, a learning algorithm can be developed that learns the scores of each category from given corpus file in accordance with review scores instead of adjusting them by hand.

DIFFICULTIES ENCOUNTERED

Since artificial intelligence algorithms desires large sets of data, preparing data is one of the time-consuming part of the project.

CONCLUSION

Briefly, artificial intelligence application is different from standard algorithms and should be thought differently. In this project, we train the system with corpus file and test it with given samples. Although wrong results are obtained in some experiments, with

more training data and better tuning of category weights, there can be much more advancements in the performance of this program.