Trabalho prático individual nº 2

Inteligência Artificial
Ano Lectivo de 2021/2022

14 de Janeiro de 2022

I Observações importantes

1. This assignment should be submitted via *GitHub* within 28 hours after its publication. The assignment can be submitted after 28 hours, but will be penalized at 5% for each additional hour.
2. Complete the requested methods in module `"tpi2.py"`, provided together with this description. Keep in mind that the language adopted in this course is Python3.
3. Include your name and number and comment or delete non-relevant code (e.g. test cases, print statements); submit only the mentioned module `"tpi2.py"`.
4. You can discuss this assignment with colleagues, but you cannot copy their programs neither in whole nor in part. Limit these discussions to the general understanding of the problem and avoid detailed discussions about implementation.
5. Include a comment with the names and numbers of the colleagues with whom you discussed this assigment. If you turn to other sources, identify those sources as well.
6. All submitted code must be original; although trusting that most students will do this, a plagiarism detection tool will be used. Students involved in plagiarism will have their submissions canceled.
7. The submitted programs will be evaluated taking into account: performance; style; and originality / evidence of independent work. Performance is mainly evaluated concerning correctness and completeness, although efficiency may also be taken into acount. Performance is evaluated through automatic testing. If necessary, the submitted modules will be analyzed by the teacher in order to appropriately credit the student's work.

II Exercícios

Together with this description, you can find modules semantic_network and bayes_net. They are similar to the ones used in practical lectures, but small changes and additions were introduced.

In particular, the class `Association` (module semantic_network) has been replaced by the following classes:

- `AssocOne` - a special kind of association where the number of values is at most one, e.g. for each entity in its first argument, the association `hasFather` accepts only one entity in its second argument (which is the father of the first argument).
- `AssocSome` - it is equivalent to the original `Association`, therefore with no limitation in the number of values.

Module `tpi2` contains some derived classes. In the following exercices, you are asked to complete certain methods in these classes. Any other code that you develop and integrate in other modules will be ignored.

The module `tpi2_tests` contains some test code, including a semantic network and a Bayesian network. You can add other test code in this module. Don't change the semantic_network and bayes_net.

You can find the intended results of `tpi2_tests` in the file `results.txt`

The responses to the main questions asked by students during this TPI will be collected in section III below.

1. Not all sources of information are equally reliable. In our semantic network, the sources are the users. Develop a method `source_confidence(user)` in class `MySemNet` that, given a user, will estimate how much confidence s/he deserves. For that purpose, the method collects all declarations of `AssocOne` relations introduced by the given user. For each entity, and for each declaraded relation, in which the entity is first argument, the method checks if the second argument is the most common one for that entity and relation, considering what the other users have declared. If the second argument is the most common one, the declaration is correct, otherwise is wrong. Based on the total numbers of correct and wrong declarations, the final confidence value is computed as follows:

$$Conf(correct, wrong) = (1 - 0.75^{correct})0.75^{wrong} \tag{1}$$

2. Due to wrong information provided by users, the content of the semantic network is also not fully reliable. Therefore, we need to estimate the confidence we can have in the obtained query results. Develop a method `query_with_confidence(entity,assoc)` in class `MySemNet` that, given an entity and the name of an `AssocOne` relation, returns a dictionary with confidence values for each possible second argument of the relation in that entity. This method performs inheritance, as was the case of the `query(entity,assoc)` method in our practical lectures. However, now we want do associate a confidence value to each alternative. The following steps should be implemented:

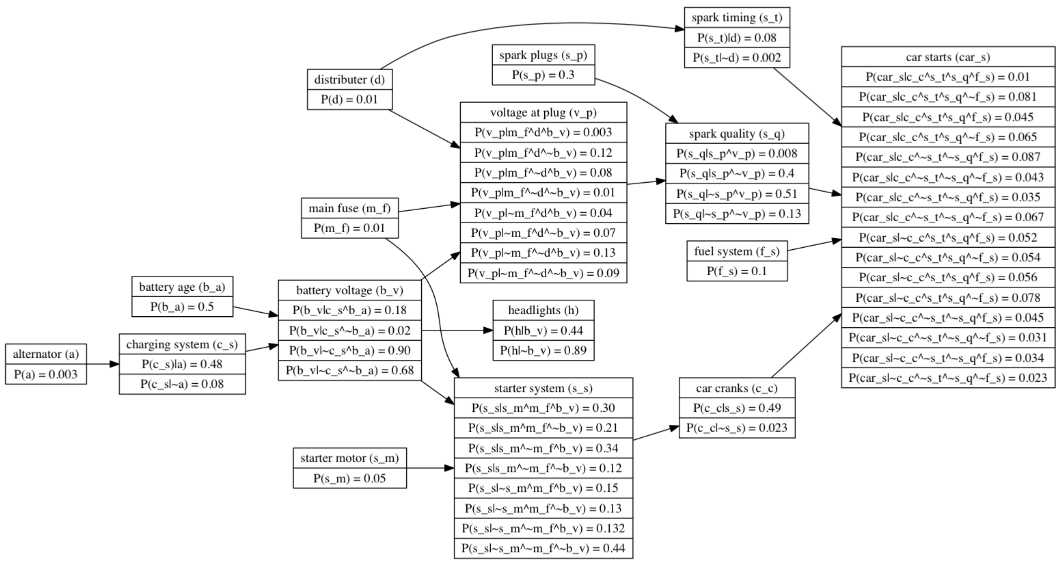
   - Compute the number of occurrences, n, for each alternative value (i.e. second argument) of `assoc` in `entity`. Compute also the total number of declarations of `assoc` in `entity`, T.
   - Compute the confidence in each value as follows:

$$Conf(n, T) = \frac{n}{2T} + \left(1 - \frac{n}{2T}\right)(1 - 0.95^{n})0.95^{T-n} \tag{2}$$

   - Call the method recursivelly for all parent entities and average the confidence results (i.e. divide by number of parents).

    - If there are no inherited results, the local results should be returned.
    - If there are no local results, the inherited results should be returned with a discount of 10% (i.e. multiply confidences by 0.9)
    - In all other cases, the final confidence values are computed by weighted average, with 0.9 for the local confidences and 0.1 for the inherited confidences.

3. Develop a method `individual_probabilities()` in class `MyBN` to compute the individual probabilities of all variables of the network. The result is given in the form of a dictionary.



spark timing (s_t)
P(s_t)|d) = 0.08
P(s_t|~d) = 0.002
spark plugs (s_p)
P(s_p) = 0.3
distributer (d)
P(d) = 0.01
car starts (car_s)
P(car_s|c_c^s_t^s_q^f_s) = 0.01
P(car_s|c_c^s_t^s_q^~f_s) = 0.081
P(car_s|c_c^s_t^s_q^f_s) = 0.045
P(car_s|c_c^s_t^s_q^~f_s) = 0.065
P(car_s|c_c^~s_t^~s_q^f_s) = 0.087
P(car_s|c_c^~s_t^~s_q^~f_s) = 0.043
P(car_s|c_c^~s_t^~s_q^f_s) = 0.035
P(car_s|c_c^~s_t^~s_q^~f_s) = 0.067
P(car_s|~c_c^s_t^s_q^f_s) = 0.052
P(car_s|~c_c^s_t^s_q^~f_s) = 0.054
P(car_s|~c_c^s_t^s_q^f_s) = 0.056
P(car_s|~c_c^s_t^s_q^~f_s) = 0.078
P(car_s|~c_c^~s_t^~s_q^f_s) = 0.045
P(car_s|~c_c^~s_t^~s_q^~f_s) = 0.031
P(car_s|~c_c^~s_t^~s_q^f_s) = 0.034
P(car_s|~c_c^~s_t^~s_q^~f_s) = 0.023
voltage at plug (v_p)
P(v_p|m_f^d^b_v) = 0.003
P(v_p|m_f^d^~b_v) = 0.12
P(v_p|m_f^~d^b_v) = 0.08
P(v_p|m_f^~d^~b_v) = 0.01
P(v_p|~m_f^d^b_v) = 0.04
P(v_p|~m_f^d^~b_v) = 0.07
P(v_p|~m_f^~d^b_v) = 0.13
P(v_p|~m_f^~d^~b_v) = 0.09
spark quality (s_q)
P(s_q|s_p^v_p) = 0.008
P(s_q|s_p^~v_p) = 0.4
P(s_q|~s_p^v_p) = 0.51
P(s_q|~s_p^~v_p) = 0.13
main fuse (m_f)
P(m_f) = 0.01
fuel system (f_s)
P(f_s) = 0.1
battery age (b_a)
P(b_a) = 0.5
battery voltage (b_v)
P(b_v|c_s^b_a) = 0.18
P(b_v|c_s^~b_a) = 0.02
P(b_v|~c_s^b_a) = 0.90
P(b_v|~c_s^~b_a) = 0.68
headlights (h)
P(h|b_v) = 0.44
P(h|~b_v) = 0.89
alternator (a)
P(a) = 0.003
charging system (c_s)
P(c_s)|a) = 0.48
P(c_s|~a) = 0.08
starter system (s_s)
P(s_s|s_m^m_f^b_v) = 0.30
P(s_s|s_m^m_f^~b_v) = 0.21
P(s_s|s_m^~m_f^b_v) = 0.34
P(s_s|s_m^~m_f^~b_v) = 0.12
P(s_s|~s_m^m_f^b_v) = 0.15
P(s_s|~s_m^m_f^~b_v) = 0.13
P(s_s|~s_m^~m_f^b_v) = 0.132
P(s_s|~s_m^~m_f^~b_v) = 0.44
car cranks (c_c)
P(c_c|s_s) = 0.49
P(c_c|~s_s) = 0.023
starter motor (s_m)
P(s_m) = 0.05

Figura 1: Bayesian network in the `tpi2_tests` module

NOTE: In this exercice, computation time will be taken into acount for the evaluation.

HINT: Instead of the method presented in the lectures, compute the individual probability of each variable only after the individual probabilities of its mother variables have been computed.

III Clarification of doubts

This work will be followed through `http://detiuaveiro.slack.com`. The clarification of the main doubts will be placed here.

1. Ex. 1 asks to look for the most common value among other users. If all values appear once, do we consider them all to be most common?

   Resposta: In case of a tie, both (or all) are correct.

2. In this part of ex. 1: "the method checks if the second argument is the most common one for that entity and relation, considering what the other users have declared." We have to consider only the other users, excluding the one passed as argument, or count the other users and the one passed as argument?

Resposta: Checks if it is the most common one, considering all declarations for that entity and relation, from all users.

3. In ex. 1, we have to consider for example feline is mammal subtype will the declared value in feline matter in the most common of mammal?

   Resposta: No, there is no bottom-up inference; it's normal inheritance.

4. Can we use imports from the standard library? (for example, methods and classes from the collections module)

   Resposta: Yes, you can use anything from the standard library.

5. In ex. 2, the n is the different values that `entity2` can have, without counting on recursion, or with recursion already?

   Resposta: n is the number of local occurrences of each value; formula (2) applies to local values; inheritance comes after the formula; you apply formula (2) for each different value of the relation.

6. In ex. 3, the `individual_probability` to be calculated for each variable is just for the boolean value `True` , correct?

   Resposta: Yes, all probabilities of being `True`.

7. In ex. 2, must all relations be of type `AssocOne`? As the function has to be recursive, if we check for the first one, we no longer allow `Member` or `SubType`.

   Resposta: The associations are `AssocOne`, but inheritance is done through the normal pathways. Ex. 2 is a very elaborate version of ex. 11a of the practical guide.

8. In ex. 2, do we have to divide by all the parents, or only by the parents that have "results"?

   Resposta: Pertinent question; my results were generated dividing by all, but I will accept both ways.

9. In ex. 3, the figure shows 0.5 for $P(b_a)$, which is not correct. What counts is what you find in the code. The figure is there essencially for you grasp more easily the topology of the network.

10. In ex. 3, the results are with the last digit sometimes different, I think it's a matter of rounding, is there a problem? For ex, instead of 0.002 I have 0.001999999999.

    Resposta: No problem with that. I'll test it so that it's not relevant.