

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 assignment. 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 account. 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In particular, the class **Association** (module `semantic_network`) has been replaced by the following classes: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 exercises, 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 declared 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:

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

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 to 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:

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

- Call the method recursively 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.

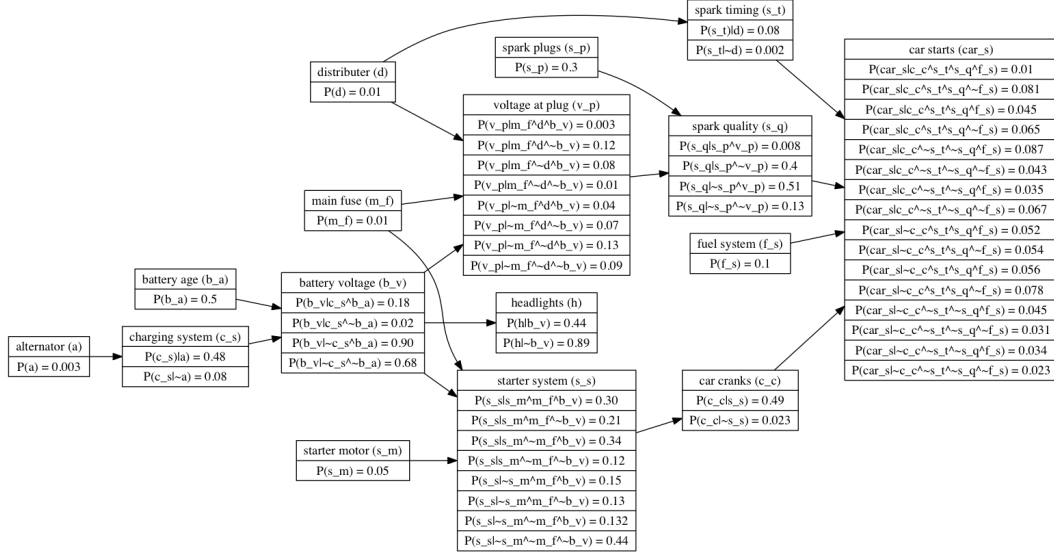


Figura 1: Bayesian network in the `tpi2.tests` module

NOTE: In this exercise, computation time will be taken into account 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.