Trabalho prático individual nº 2

Inteligência Artificial
Ano Lectivo de 2023/2024

8 de Dezembro de 2023

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 these 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 constraintsearch . They are similar to the ones used in practical lectures, but some changes and additions were introduced.

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. You can add other test code in this module. Don't change the semantic_network and constraintsearch modules.

You can find the intended results of `tpi2_tests` in the file `tpi2_results.txt`

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

1. The attached module `semantic_network` was designed based on the one you already know, from practical classes, but has some changes were introduced:

   - All network data is know in the form of a dictionary of dictionaries (see code).
   - The following convention was introduced: object names start with upper case letter (e.g. 'Mary') and type names start with lower case letter (e.g. 'woman'). Convenience functions are provided to check if a name is an object or type.
   - There is a new type of association, `AssocOne`, which allows only one value (e.g. hasMother: each person has only one mother).

   You are expected to implement the following methods in class `MySN`

   a) `query_local(user,e1,rel,e2)` - Equivalent to the method with the same name that you know from practical classes.

   b) `query(entity,assoc)` - Equivalent to the method with the same name that you developed in practical classes.

   c) `update_assoc_stats(assoc,user=None)` - This method stores and/or updates frequency statistics in a dictionary where keys are tuples `(assoc,user)` and values are tuples `(stats_assoc_e1,stats_assoc_e2)`, i.e. frequency statistics for the types of the first and second arguments of the association `assoc`. If `user==None`, the statistics are computed based on the declarations of all users. Otherwise, the statistics are computed based on the declarations of the specified user. For each of the arguments, the method will collect all objects used in that argument, in the available declarations, check their types, compute frequencies of occurrence of each type, and propagate the information upwards in the type hierarchy. For `N` such objects, where `K` of them are of unknown type, the total number of objects to be used for calculating relative frequencies is given by `N-K+K**(1/2)`. See the results file for examples and details.

2. Implement in class `MyCS` the method `search_all()`. Instead of a normal `search()` method, this one will return all possible solutions without repetitions. This exercice will be evaluated based on correctness, completeness and time efficiency. In order to improve efficiency, consider including constraint propagation and some form of variable ordering.

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. Podemos adicionar argumentos às funções?

   Resposta: Only optional arguments located after the argments already included; for testing, I will call the functions with the arguments that are identified in the provided code, and no others.

2. O `query_local` também tem de funcionar com o `AssocOne`?

   Resposta: Sim!

3. Poderia explicar o que quer dizer com isto: "For N such objects, where K of them are of unknown type, the total number of objects to be used for calculating relative frequencies is given by N-K+K**(1/2). See the results file for examples and details"?

   Resposta: Para os objectos que aparecem, por exemplo, no 1º argumento (`e1`) da associação dada, alguns ainda não têm tipo declarado pelo `user`, portanto são de tipo desconhecido; para o restantes, vai ver quais os tipos e calcula a percentagem de ocorrência (frequência relativa), mas em vez de dividir por N, divide por N-K+K**(1/2).

4. Relativamente às cotações? Como é que estão distribuídas?

   Resposta: Ainda não tenho cotações, mas vão reflectir a quantidade de trabalho bem como a dificuldade: 1.a) quantidade de trabalho / dificuldade média; 1.b) quantidade de trabalho / dificuldade baixa; 1.c) e 2. quantidade de trabalho e dificuldade mais elevadas. As verdadeiras cotações só acabam por ficar definidas quando for implementado o programa de avaliação automática; não são atribuídas cotações a exercícios, mas sim a testes, e os testes por vezes reflectem o funcionamento de mais do que um exercício.

5. Consegue explicar o resultado obtido no 1.b) por favor?

   ```
   Socrates:  [decl(Darwin,likes(man,meat))]
   ```

   Resposta: Se pensarem bem, está coerente com a forma como funciona a rede semântica das aulas. O `Socrates` herda `"likes meat"` de `man`.

6. Porque é que, nesta adição (linhas 67 a 70 dos resultados):

   ```
   ( {'mammal': 0.5, 'woman': 0.5, 'human': 0.5, 'vertebrate': 0.5},
     {'mammal': 0.5, 'woman': 0.5, 'human': 0.5, 'vertebrate': 0.5} )

   Added: decl(Darwin,hasMother(human,woman))

   ( {'mammal': 0.5, 'woman': 0.5, 'human': 0.5, 'vertebrate': 0.5},
     {'mammal': 0.5, 'woman': 0.5, 'human': 0.5, 'vertebrate': 0.5} )
   ```

   não mudam nada nos resultados? (o `update_assoc_stats()` e o `print()` são ambos feitos com `None`).

   Resposta: As estatísticas baseiam-se exclusivamente nos objectos, e essa relação `hasMother(human,woman)` é uma relação entre tipos. Ou seja, não contribuiu para a estatística. Vocês podem ver estas estatísticas como uma fonte de informação complementar (poderia por exemplo ter um `query()` mais elaborado que, não tendo declaração explícita dos tipos dos argumentos [é o que nos diz `hasMother(human,woman)`], poderia recorrer às estatísticas para fazer algumas inferências.

7. Nos resultados (principalmente o `assoc_stats`), os resultados precisam de estar por ordem. E se sim, que critério de ordenação devemos usar?

   Resposta: Não precisam estar por ordem!

8. Quanto é um tempo "bom"no exercício 2.?

   Resposta: O meu algoritmo/implementação no meu computador demora menos de 0.01 segundos.

9. No ex 1.b, no `query()` é só para devolver as associações herdadas por `member`, ou também as por `subtype`? E são só as herdadas diretamente, ou seja, as do parente, ou também aquelas que este já herdava?

   Resposta: O `query()` no ex. 1.b) tem exactamente a mesma funcionalidade do `query()` das aulas: associações herdadas via `member` e `subtype`, sem cancelamento de herança.

10. Pode darnos mais testes para a 1.b em questao?

    Resposta: Sugere-se que utilizem o módulo `semantic_network` original, com o `query()` implementado nas aulas; o resultado tem que ser igual ao `query()` pretendido hoje ... Com o `query()` das aulas, geram quantos casos de teste quiserem. Fazem um cópia do `tpi2_tests` para a pasta das aulas, substituem o `import tpi2` por `import semantic_network`, tiram as chamadas às funções 1.c) e 2. e vêm o que acontece com o resto; os resultados do `query_local()` e `query()` devem ser iguais.

11. No exercício 1.c) este teste `z.update_assoc_stats('teacher')` está correto? (linha 66 do `tpi2_tests.py`). Há 6 associations do tipo `teacher`:

    ```
    z.insert(Declaration('Descartes',Association('Socrates','teacher','Philosophy')))
    z.insert(Declaration('Descartes',Association('Socrates','teacher','Mathematics')))
    z.insert(Declaration('Descartes',Association('Plato','teacher','Philosophy')))
    z.insert(Declaration('Yesao',Association('Socrates','teacher','Mathematics')))
    z.insert(Declaration('Yesao',Association('Plato','teacher','Philosophy')))
    z.insert(Declaration('Descartes',Association('Elvira','teacher','Philosophy')))
    ```

    A frequência de `discipline` (apenas `philosophy` é do tipo `discipline`) para o user `None`, nao devia ser então $4/6 = 0.66$, em vez do valor que está no `tpi2_results.txt` que é 0.7387961250362586?

    Resposta: Está a ignorar alguns pormenores do enunciado. Ver também a pergunta nº 3 acima.

12. Podemos adicionar mais argumentos no `search_all()` ou apenas podemos usar o `xpto` ?

    Resposta: Sim, mas apenas argumentos opcionais.

13. Podemos usar o `defaultdict` de `collections.abc` para o `assoc_stats` do exercício 1.c)?

    Resposta: Se isso originar problemas de compatibilidade com o meu código de avaliação, fica à vossa responsabilidade.

14. De preferência, não usem funcionalidades do Python posteriores ao Python 3.10, que é a versão mais recente que tenho instalada, e que estou a pensar usar na avaliação.

15. Há problema de o output ficar assim para a função `query()`:

    `Elvira: [decl(Darwin,breastfeed(mammal,'Yes'))]`

    em vez de assim: `Elvira: [decl(Darwin,breastfeed(mammal,Yes))]`

    Resposta: Por acaso essa associação devia estar introduzida como `AssocOne`; no entanto, como está introduzida como `Association`, isso implica desdobrar o conjunto de valores existentes e declarações independentes; é fazer o inverso do que faz o `insert()` quando se trata de uma `Association`

16. Alguns alunos estão a cometer erros básicos: as relações de `subtype` servem como caminho de herança, mas elas próprias não são herdáveis. Não faz sentido dizer que, por herança de `human`, o `Socrates` seja subtipo de `mammal`.

17. Quando fazemos `query` com `Socrates` sem especificar associação, não devia aparecer a relação em que o `Socrates` é membro de `man`?

    Resposta: O `query()` só vai buscar associações; segundo as definições que sempre usámos, as relações de `member` e `subtype` não são associações.