RELATÓRIO DO SPRINT 3

Turma 2DH _ Grupo 43
1190929 _ Patrícia Barbosa
1190947 _ Pedro Fraga
1190956 _ Pedro Garcia
1190963 _ Pedro Preto

Professora:
Brígida Teixeira, BCT

Unidade Curricular:
Algoritmia Avançada

Data: 23/01/2022

ÍNDICE

ÍNDICE DE FIGURAS

Considerar estados emocionais para encontrar os caminhos:

```
                                %joy, hope, relief, pride, gratitude
no(1,ana,[natureza,pintura,musica,sw,porto],0.5,0.7,0.8,0.9,1).
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.2,0.3,0.4,0.5,0.1).
no(12,beatriz,[natureza,musica,carros,porto,moda],0.4,0.4,0.6,0.5,0.3).
```

Figura 1 – Base de conhecimento Sprint D com estados emocionais

No novo sprint começamos por adicionar novos dados à base de conhecimento como se pode verificar na figura 1. Estes dados, são relativos aos estados emocionais, sendo que só foram adicionados os estados com conotação positiva, uma vez que decidimos seguir a seguinte lógica: se um utilizador está “0.6” contente, significa que está “0.4” triste.

```
bestfs1(Orig,Dest,MaximoLigacoes,Cam,Custo):-
asserta(caminho_somatorio([],0)),
bestfs12(Dest,[[Orig]],MaximoLigacoes,Cam,Custo),
    write('Caminho-'),write(Cam),nl,!.

bestfs12(Dest,[[Dest|T]|_],_,Cam,Custo):- reverse([Dest|T],Cam),
    calcula_custo(Cam,Custo).

bestfs12(Dest,[[Dest|_]|LLA2],MaximoLigacoes,Cam,Custo):- !, bestfs12(Dest,LLA2,MaximoLigacoes,Cam,Custo).

bestfs12(Dest,LLA,MaximoLigacoes,Cam,Custo):- member1(LA,LLA,LLA1), LA=[Act|_],
    ((Act==Dest,!,bestfs12(Dest,[LA|LLA1],MaximoLigacoes,Cam,Custo))
    ;
    (findall((CX,[X|LA]),(edge(Act,X,CX), \+member(X,LA)),Novos),
    Novos\==[],!,
    sort(0,@>=,Novos,NovosOrd),
    retira_custos(NovosOrd,NovosOrd1),
    append(NovosOrd1,LLA1,LLA2),
    write('============================================='),nl,
    write('LLA2='),write(LLA2),nl,
    retrack_paths_with_max_connections(LLA2,MaximoLigacoes,LLA2V2),
    write('LLA2V2='),write(LLA2V2),nl,
    retrack_paths_with_unallowed_emotional_state(LLA2V2,LLA2V3),
    write('LLA2V3='),write(LLA2V3),nl,
    write('============================================='),nl,
```

Figura 2 - bestf1/5: Best First adaptado de modo a considerar os estados emocionais

O método alterado para considerar estados emocionais foi o “Best First” (Figura 2),e a única alteração que foi feita foi adicionar um novo predicado ”retrack_paths_with_unallowed_emotional_state” (Figura 3).

```
retrack_paths_with_unallowed_emotional_state([],[]).
retrack_paths_with_unallowed_emotional_state([Head|Tail],[Head|Lnew]):-
      check_emotional_state(Head),!,
      retrack_paths_with_unallowed_emotional_state(Tail,Lnew).
retrack_paths_with_unallowed_emotional_state([_|Tail],Lnew):-
        retrack_paths_with_unallowed_emotional_state(Tail,Lnew).
```

Figura 3 - retrack_paths_with_unallowed_emotional_state/2

O predicado da Figura 3, vai receber o uma lista de listas com todos os caminhos até n ligações, e posteriormente vai mandar cada um dos caminhos da lista para o predicado check_emotional_state (Figura 4), que vai verificar se o caminho é permitido. Caso seja, adiciona o caminho à lista de listas de retorno, caso contrário passa para o caminho seguinte.

```
check_emotional_state([]).
check_emotional_state([Head|Tail]):-
no(_,Head,_,J,H,R,P,G),J>=0.5,H>=0.5,R>=0.5,P>=0.5,G>=0.5,
check_emotional_state(Tail).
```

Figura 4 - check_emotional_state/1

O método da Figura 4, recebe um caminho, e vai percorrer todos os seus utilizadores e consequentemente verificar se os seus estados emocionais são maiores ou iguais a 0.5. Se esta condição for verificada retorna “true”, caso contrário retorna false.

Demonstração:

```
?- bestfs1(ana,preto,9,Caminho,Custo).
```

Figura 5 - execução do predicado

Se no executarmos o predicado da Figura 5, o resultado vai ser o seguinte (Figura 6):

```
Caminho = [ana, daniel, eduardo, maria, patricia, preto],
Custo = 21.
```

Figura 6 - Resultado do predicado da figura 5

Porém, de modo a verificar-mos que o predicado “Best First” adaptado a estados emocionais (Figura 2), realmente funciona, foi alterado na base de conhecimentos o estado emocional hope do utilizador “patricia” para 0.4. Deste modo, o predicado da Figura 2, não deve considerar nenhum caminho que tenha este utilizador.

```
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.4,0.5,0.5,0.5).
```

Figura 7 - Alteração na base de conhecimento

Deste modo, voltou-se a executar o predicado da Figura 5, e obteve-se o resultado da Figura 8, e como se pode verificar o caminho retornado já não inclui o utilizador “patricia”

```
Caminho = [ana, daniel, eduardo, maria, garcia, preto],
Custo = 15.
```

Figura 8 - Novo resultado do predicado da figura 5

Sugestão de Grupos:

Para responder a este requisito foi desenvolvido o predicado sugestão_grupos/5, que consiste na sugestão do maior grupo para um dado utilizador desde que tenha um mínimo de N utilizadores e T tags comuns, considerando que algumas tags podem ser obrigatórias.

Em primeiro lugar chama-se o predicado todas_combinacoes/3 que devolve as tags agrupadas T a T. A lista de lista de tags combinadas devolvida é enviada ao predicado verifica_tags_obrigatorias/4 juntamente com a lista de tags obrigatórias e vai verificar e manter na lista apenas as combinações de tags que contenham as tags obrigatórias. Posteriormente, através do findall/3, guarda-se todos os nós da base de conhecimento numa lista e envia-se juntamente com a lista de tags combinadas e obrigatórias ao predicado selecionaCombTags/3 que vai para cada lista de tags ver que users é que partilham das mesmas.

```
/*  Predicado principal   */
    /*  LTags - Lista de tags
        T - Numero de tags em comum
        N - Numero minimo de users no grupo sugerido
        LTagsObrigatorias - lista de tags obrigatorias
        LUsersComCombTags - lista de users agrupados por T tags em comum e obrigatorias - RETORNO
    */
sugestao_grupos(LTags,T, N, LTagsObrigatorias, LUsersComCombTags):-
    todas_combinacoes(T,LTags,LcombXTags),
    write('LTagsComb: '), write(LcombXTags),nl,nl,
    verifica_tags_obrigatorias(LcombXTags, LTagsObrigatorias, T, LTagsCombEObrigatorias),
    write('LTagsCombEObrigatorias: '), write(LTagsCombEObrigatorias),nl,nl,
    findall(X, no(X,_,_,_,_,_,_,_), LUsers),
    selecionaCombTags(LTagsCombEObrigatorias,LUsers,LUsersComCombTags).
```

Figura 9 - sugestao_grupos/5

O predicado verifica_tags_obrigatorias/4, vai extraindo lista a lista da lista de tags combinadas e para cada uma delas faz a interseção com a lista de tags obrigatórias, atrasves do predicado intersecao/3, e verifica se o tamanho da lista interseção é igual ao tamanho da lista de tags obrigatórias (esta condição é uma forma de verificar se as tags obrigatórias estão na combinação de tags), atrav´s do predicado lenght/2. Se o tamanho for igual, chama recursivamente o atual predicado, adicionando essa lista de tags à LTagsCombEObrigatorias, caso contrário chama recursivamente o predicado atual sem acrescentar nada ao último parâmetro. A condição de paragem da recursividade deste método é a lista de tags combinadas encontrar-se vazia, o que significa que não existe mais verificações a fazer.

```
/*  Verifica se as combinações de tags tem as tags obrigatorias   */
    /*  LTagsComb - Lista de tags combinadas T a T, resultado de todas_combinacoes
        LTagsObrigatorias - Lista de tags obrigatorias
        T - numero de tags obrigatorias
        LTagsCombEObrigatorias - Lista de tags combinadas T a T que contem as tags obrigatorias - RETORNO
    */
verifica_tags_obrigatorias([],_,_,LT):-!.
verifica_tags_obrigatorias([X|LTagsComb], LTagsObrigatorias, T, LTagsCombEObrigatorias):-
    intersecao(X, LTagsObrigatorias, LR),
    length(LTagsObrigatorias,Referencia),
    length(LR,Length),
    %write('LTagsComb: '), write(LTagsComb),nl,
    %write('Intersecao: '), write(LR),nl,
    %write('Length: '), write(Length),nl,
    %write('X: '), write(X),nl,
    (Length == Referencia -> verifica_tags_obrigatorias(LTagsComb,LTagsObrigatorias,T,LFinal),
                                LTagsCombEObrigatorias = [X|LFinal];
                            verifica_tags_obrigatorias(LTagsComb,LTagsObrigatorias,T,LTagsCombEObrigatorias)).
    %write("LTagsCombEObrigatorias: "), write(LTagsCombEObrigatorias),nl,
    %write("LFINAL: "), write(LFinal),nl,nl,
    %verifica_tags_obrigatorias(LTagsComb,LTagsObrigatorias,T,LFinal).
```

Figura 10 - verifica_tags_obrigatorias/4

O predicado selecionaCombTags/3, para cada combinação de tags vai verificar todos os users e confirmar quais deles as contém, para isso é utlizado o predicado comparaComUsers/3. Este método é chamado recursivamente até LT ser vazio, o que significa que se verificou todas as combinações de tags.

```
/*  Itera sobre a lista de tags combinadas T a T que contem as tags obrigatorias e para cada uma delas chama
    o metodo que verifica os users que as contêm   */
    /*  LT - Lista de tags combinadas T a T que contem as tags obrigatorias
        LUsers - Lista com os ids de todos os users que estao na base de connhecimento
        LUsersComCombTags - Lista com listas de users que partilham de cada combinação de tags em comum e
            obrigatorias - RETORNO FINAL!!!
    */
selecionaCombTags([],_,LUsersComCombTags):-!.
selecionaCombTags([Lista|LT], LUsers,LUsersComCombTags):-
    comparaComUsers(Lista, LUsers, LAux),
    write(Lista), write(": "), write(LAux),nl,
    selecionaCombTags(LT,LUsers, LUsersComCombTags), LL =[LAux|LUsersComCombTags].
```

Figura 11- selecionaCombTags/3

O predicado comparaComUsers/3 para cada user da lista de users passada por parâmetro verifica se a lista de tags desse user contem a lista de tags que esta a ser verificada (Lista), para isso recorre-se novamente ao predicado interseção e posterior verificação do tamanho da lista resultado. Caso esta tenha o mesmo tamanho que a Lista a ser verificada, significa que o user em questão tem pelo menos aquelas tags em comum e então chama recursivamente o atual predicado acrescentando à lista final esse user. Caso não se verifique apenas é chamado recursivamente o atual predicado, mas sem acrescentar nada à lista final. A condição de paragem da recursividade deste método é a lista de users estar vazia pois significa que já se comparou com todos eles.

```
/*  Verifica para cada lista de tags os users que as têm   */
    /*  Lista - Lista de tags
        LUsers -  Lista com os ids de todos os users que estao na base de connhecimento
        LUsersComCombTags - Lista com listas de users que partilham de cada combinação de tags em comum
            e obrigatorias - RETORNO FINAL!!!
    */
comparaComUsers(Lista,[],L):-!.
comparaComUsers(Lista, [User|LUsers],LUsersComCombTags):-
    no(User, _,TagsUser,_,_,_,_,_),
    %write("User: "), write(User),nl,
    %write("Lista:"), write(Lista),nl,
    %write("TagsUser:"), write(TagsUser),nl,
    intersecao(Lista, TagsUser, LR),
    %write("Lista intersecao: "), write(LR),nl,
    length(Lista,Referencia),
    length(LR,Compara),
    %write("REFERENCIA: "), write(Referencia), write("   COMPARA: "), write(Compara),nl,
    (Referencia == Compara -> comparaComUsers(Lista, LUsers, LFinal), LUsersComCombTags = [User|LFinal];
                              comparaComUsers(Lista, LUsers, LUsersComCombTags)).
```

Figura 12- comparaComUsers/3

Como predicados auxiliares foi utilizado o todas_combinacoes/3, fornecido nas aulas, para calcular todas as combinações de tags X a X, e ainda o predicado interseção, também fornecido nas aulas, para calcular a interseção de duas listas.

```
/*  PREDICADOS AUXILIARES - FORNECIDOS  */
todas_combinacoes(X,LTags,LcombXTags):-findall(L,combinacao(X,LTags,L),LcombXTags).
combinacao(0,_,[]):-!.
combinacao(X,[Tag|L],[Tag|T]):-X1 is X-1, combinacao(X1,L,T).
combinacao(X,[_|L],T):- combinacao(X,L,T).

intersecao([ ],_,[ ]).
intersecao([X|L1],L2,[X|LI]):-member(X,L2),!,intersecao(L1,L2,LI).
intersecao([_|L1],L2, LI):- intersecao(L1,L2,LI).
```

Figura 13 - Predicados auxiliares

Cálculo de novos valores dos estados emocionais:

No final de todos os métodos implementados, a base de conhecimento de utilizadores é atualizada com os novos valores, portanto, todos terminam da seguinte forma:

> EmocaoEmQuestao is (fórmula para calcular o novo valor),
> asserta(no(UserId,A,B,C,D,E,F,G,H)), //(trocando o valor da emocao em questao com a variavel EmocaoEmQuestao)
> save(). //método para atualizar a base de conhecimento

```
save():-
  getWorkingDirectory(Dir),
  concat(Dir, 'usersBC.pl', FilePath),
  write_facts(FilePath).
```

Figura 14 - save

O save envia para o método write_facts o ficheiro pretendido (a base de conhecimento pretendida), para isto é necessário alterar o diretório em getWorkingDirectory:

```
getWorkingDirectory('C:/Users/topel/Documents/1apr5_g43/projeto base exemplo/nunopsilva-dddnetcore-c2122b82f1b9/ALGAV/Sprint D/').
```

Figura 15 - getWorkingDirectory/1

```
write_facts(FileName):-
  open(FileName,write,Out),
  forall(
  no(A,B,C,D,E,F,G,H,I),(
  write(Out,"no("),
  write(Out,A),write(Out,","),
  write(Out,B),write(Out,","),
  write(Out,C),write(Out,","),
  write(Out,D),write(Out,","),
  write(Out,E),write(Out,","),
  write(Out,F),write(Out,","),
  write(Out,G),write(Out,","),
  write(Out,H),write(Out,","),
  write(Out,I),write(Out,")"),
  writeln(Out,"."))),
  close(Out).
```

Figura 16 - write_facts/1

O write facts escreve na base de conhecimento todos os nós, alterando apenas o nó em que foi modificado um estado emocional, pois a este nó é feito retract e assert com os novos valores.

Os estados emocionais são os seguintes:

```
% Joy -> Like/Dislike ratio
% Relief -> Friends in groups
% Pride -> Tags same as friends'
% Gratitude -> Tags same as groups'
```

Figura 17 - estados emocionais e respetivos parâmetros/dependências

São apenas calculados os valores dos estados emocionais ”positivos”, visto que os valores dos mesmos variam entre 0 e 1, portanto os valores dos estados emocionais opostos, assumindo entao que o valor do estado emocional negativo é 1 – o valor do positivo. Os valores de Relief, Pride e Gratitude tem em comentário uma forma alternativa do calculo do seu valor.

Cálculo da felicidade (Joy - 4º parâmetro do predicado no):

```
calculate_joy(UserId):-
  retract(no(UserId,A,B,InitialUserJoy,C,D,E,F,LikeDifference)),
  UserJoy is InitialUserJoy + (1 - InitialUserJoy) * (min(LikeDifference, 200) / 200),
  asserta(no(UserId,A,B,UserJoy,C,D,E,F,LikeDifference)),
  save(),
  write('User Joy changed from '), write(InitialUserJoy), write(' to '), write(UserJoy).
```

Figura 18 - calculate_joy/1

Como a felicidade depende da diferença entre os likes e os dislikes nos posts do utilizador, com um valor de saturação de 200, o método simplesmente procura o nó do utilizador com o id recebido e guarda o valor inicial da sua felicidade, dando lhe retract, e utilizando-o para calcular o novo valor, depois é feito o assert com o novo valor de felicidade.

Exemplo:

Base de conhecimento inicial:

```
no(1,ana,[natureza,pintura,musica,sw,porto],0.999996875,0.7,0.3333333333333333,0.5652173913043478,0.6,190).
no(200,preto,[natureza,bolachas,moda,musica,sw,coimbra],0.565,0.5,0,0.4166666666666667,0.5,26).
no(21,eduardo,[natureza,cinema,teatro,carros,coimbra],0.78975,0.9,0.9,0.9,0.9,-90).
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.276,0.3,0.4,0.5,0.1,19).
no(12,beatriz,[natureza,musica,carros,porto,moda],0.4,0.4,0.6,0.5,0.3,90).
no(13,carlos,[natureza,musica,sw,futebol,coimbra],0.1,0.2,0.3,0.1,0,10).
no(14,daniel,[natureza,cinema,jogos,sw,moda],0.8,0.8,0.8,0.6,0.8,1).
no(22,isabel,[natureza,musica,porto,lisboa,cinema],0.8,0.8,0.8,0.8,0.8,100).
no(23,jose,[natureza,pintura,sw,musica,carros,lisboa],0.7,0.7,0.7,0.7,0.8,80).
no(24,luisa,[natureza,cinema,jogos,moda,porto],0.6,0.6,0.6,0.6,0.6,112).
no(31,maria,[natureza,pintura,musica,moda,porto],0.5,0.5,0.5,0.5,0.5,312).
no(32,anabela,[natureza,cinema,musica,tecnologia,porto],0.4,0.4,0.4,0.4,0.4,51).
no(33,andre,[natureza,carros,futebol,coimbra],0.3,0.3,0.3,0.3,0.3,42).
no(34,catia,[natureza,musica,cinema,lisboa,moda],0.2,0.2,0.2,0.2,0.2,45).
no(41,garcia,[natureza,teatro,tecnologia,futebol,porto,bolachas],0.5,0.5,0.5,0.5,0.5,35).
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.5,0.5,0.5,0.5,78).
no(43,fraga,[natureza,teatro,carros,porto,bolachas],0.5,0.5,0.5,0.5,0.5,73).
no(44,isaura,[natureza,moda,tecnologia,cinema],0.2,0.3,0.4,0.4,0.3,150).
no(51,rodolfo,[natureza,musica,sw],0.2,0.3,0.4,0.5,0.4,137).
no(61,rita,[moda,tecnologia,cinema],0.2,0.3,0.4,0.5,0.5,-5).
```

Figura 19 – Base de conhecimentos antes da execução do predicado calculate_joy/1

Cálculo do novo valor de felicidade para o utilizador 12:

```
?- calculate_joy(12).
User Joy changed from 0.4 to 0.67
true.
```

Figura 20 - execução na consola do predicado calculate_joy/1

Base de conhecimento depois de calcular o novo valor:

```
no(12,beatriz,[natureza,musica,carros,porto,moda],0.67,0.4,0.6,0.5,0.3,90).
no(1,ana,[natureza,pintura,musica,sw,porto],0.999996875,0.7,0.3333333333333333,0.5652173913043478,0.6,190).
no(200,preto,[natureza,bolachas,moda,musica,sw,coimbra],0.565,0.5,0,0.4166666666666667,0.5,26).
no(21,eduardo,[natureza,cinema,teatro,carros,coimbra],0.78975,0.9,0.9,0.9,0.9,-90).
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.276,0.3,0.4,0.5,0.1,19).
no(13,carlos,[natureza,musica,sw,futebol,coimbra],0.1,0.2,0.3,0.1,0,10).
no(14,daniel,[natureza,cinema,jogos,sw,moda],0.8,0.8,0.8,0.6,0.8,1).
no(22,isabel,[natureza,musica,porto,lisboa,cinema],0.8,0.8,0.8,0.8,0.8,100).
no(23,jose,[natureza,pintura,sw,musica,carros,lisboa],0.7,0.7,0.7,0.7,0.8,80).
no(24,luisa,[natureza,cinema,jogos,moda,porto],0.6,0.6,0.6,0.6,0.6,112).
no(31,maria,[natureza,pintura,musica,moda,porto],0.5,0.5,0.5,0.5,0.5,312).
no(32,anabela,[natureza,cinema,musica,tecnologia,porto],0.4,0.4,0.4,0.4,0.4,51).
no(33,andre,[natureza,carros,futebol,coimbra],0.3,0.3,0.3,0.3,0.3,42).
no(34,catia,[natureza,musica,cinema,lisboa,moda],0.2,0.2,0.2,0.2,0.2,45).
no(41,garcia,[natureza,teatro,tecnologia,futebol,porto,bolachas],0.5,0.5,0.5,0.5,0.5,35).
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.5,0.5,0.5,0.5,78).
no(43,fraga,[natureza,teatro,carros,porto,bolachas],0.5,0.5,0.5,0.5,0.5,73).
no(44,isaura,[natureza,moda,tecnologia,cinema],0.2,0.3,0.4,0.4,0.3,150).
no(51,rodolfo,[natureza,musica,sw],0.2,0.3,0.4,0.5,0.4,137).
no(61,rita,[moda,tecnologia,cinema],0.2,0.3,0.4,0.5,0.5,-5).
```

Figura 21 - Base de conhecimentos depois da execução do predicado calculate_joy/1

Como se pode verificar, o no 12, que foi o selecionado, passou para primeiro na base de conhecimento, isto acontece porque foi o no a ser alterado, podemos também verificar que o valor da sua felicidade passou de 0.4 para 0.67, como indicado na execução do programa.

Cálculo do alívio (Relief - 6º parâmetro do predicado no):

```
calculate_relief(UserId):-
  friend_list_of_user(UserId, MyFriends),
  users_in_groups_of_user(UserId, UsersInTheSameGroupsAsMe),
  flatten(UsersInTheSameGroupsAsMe, FlattenedUserList),
  sort(FlattenedUserList, PeopleInMyGroups),
  length(PeopleInMyGroups, PeopleInMyGroupsLength),
  intersection(MyFriends, PeopleInMyGroups, FriendsInMyGroups),
  length(FriendsInMyGroups, FriendsInMyGroupsLength),
  retract(no(UserId,A,B,C,D,InitialUserRelief,E,F,G)),
  % UserRelief is InitialUserRelief + (1 - InitialUserRelief) * (FriendsInMyGroupsLength / max(People
  UserRelief is (FriendsInMyGroupsLength / max(PeopleInMyGroupsLength, 1)),
  asserta(no(UserId,A,B,C,D,UserRelief,E,F,G)),
  save(),
    write('User Relief changed from '), write(InitialUserRelief), write(' to '), write(UserRelief).
```

Figura 22 - calculat_relief/1

Como o alívio depende da percentagem do número de amigos do utilizador em grupos do mesmo em relação ao número total de utilizadores nos grupos do mesmo, o método primeiro procura os amigos do utilizador com o predicado friend_list_of_user:

```
friend_list_of_user(UserId, FriendList):-
  findall(FriendId, friend(UserId, FriendId), FriendList).
```

Figura 23 - friend_list_of_user/2

De seguida, procura todos os utilizadores em grupos em que o utilizador selecionado se encontra com o predicado users_in_groups_of_user:

```
users_in_groups_of_user(UserId, UserList):-
  findall(UsersOfGroup, grupo(UsersOfGroup,_), ListOfUsersOfGroups),
  concat_lists_user_is_in(UserId, ListOfUsersOfGroups, UserList).
```

Figura 24 - users_in_groups_of_users/2

Este predicado primeiro procura todos os grupos e guarda em ListOfUsersOfGroups todas as listas de utilizadores de todos os grupos, de seguida chama o predicado concat_lists_user_is_in:

```
concat_lists_user_is_in(_, [], []):- !.
concat_lists_user_is_in(UserId, [ActualList|ListOfLists], ListOfCommon):-
  (member(UserId, ActualList) ->
  concat_lists_user_is_in(UserId, ListOfLists, RestOfCommon), ListOfCommon = [ActualList|RestOfCommon];
  concat_lists_user_is_in(UserId, ListOfLists, ListOfCommon)).
```

Figura 25 - concat_lists_user_is_in/3

Este predicado verifica se o utilizador em questão se encontra na lista de utilizadores de cada grupo, se sim, guarda a lista de utilizadores desse grupo em outra lista.

Voltando ao predicado principal, depois de executar o users_in_groups_of_user, ele junta todas as listas da lista recebida em uma única lista com o flatten e remove os duplicados com o sort, obtendo então a lista de todos os utilizadores em grupos do utilizador selecionado.
De seguida é feita a interseção entre a lista de amigos do utilizador e a lista de utilizadores dos grupos do mesmo, obtendo, então, a lista de amigos que estão em grupos do utilizador selecionado.
É feito o retract dos valores do no do utilizador selecionado, o novo valor da UserRelief é calculado com, obtendo então a percentagem do número de amigos do utilizador em grupos do mesmo em relação ao número total de utilizadores nos grupos do mesmo. É feito o assert dos novos valores e os resultados são guardados com o save.

Exemplo:

Base de conhecimento inicial:

```
no(12,beatriz,[natureza,musica,carros,porto,moda],0.67,0.4,0.6,0.5,0.3,90).
no(1,ana,[natureza,pintura,musica,sw,porto],0.999996875,0.7,0.3333333333333333,0.5652173913043478,0.6,190).
no(200,preto,[natureza,bolachas,moda,musica,sw,coimbra],0.565,0.5,0,0.4166666666666667,0.5,26).
no(21,eduardo,[natureza,cinema,teatro,carros,coimbra],0.78975,0.9,0.9,0.9,0.9,-90).
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.276,0.3,0.4,0.5,0.1,19).
no(13,carlos,[natureza,musica,sw,futebol,coimbra],0.1,0.2,0.3,0.1,0,10).
no(14,daniel,[natureza,cinema,jogos,sw,moda],0.8,0.8,0.8,0.6,0.8,1).
no(22,isabel,[natureza,musica,porto,lisboa,cinema],0.8,0.8,0.8,0.8,0.8,100).
no(23,jose,[natureza,pintura,sw,musica,carros,lisboa],0.7,0.7,0.7,0.7,0.8,80).
no(24,luisa,[natureza,cinema,jogos,moda,porto],0.6,0.6,0.6,0.6,0.6,112).
no(31,maria,[natureza,pintura,musica,moda,porto],0.5,0.5,0.5,0.5,0.5,312).
no(32,anabela,[natureza,cinema,musica,tecnologia,porto],0.4,0.4,0.4,0.4,0.4,51).
no(33,andre,[natureza,carros,futebol,coimbra],0.3,0.3,0.3,0.3,0.3,42).
no(34,catia,[natureza,musica,cinema,lisboa,moda],0.2,0.2,0.2,0.2,0.2,45).
no(41,garcia,[natureza,teatro,tecnologia,futebol,porto,bolachas],0.5,0.5,0.5,0.5,0.5,35).
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.5,0.5,0.5,0.5,78).
no(43,fraga,[natureza,teatro,carros,porto,bolachas],0.5,0.5,0.5,0.5,0.5,73).
no(44,isaura,[natureza,moda,tecnologia,cinema],0.2,0.3,0.4,0.4,0.3,150).
no(51,rodolfo,[natureza,musica,sw],0.2,0.3,0.4,0.5,0.4,137).
no(61,rita,[moda,tecnologia,cinema],0.2,0.3,0.4,0.5,0.5,-5).
```

Figura 26 - Base de conhecimentos antes da execução do predicado calculate_relief/2

Cálculo do novo valor de alívio para o utilizador 11:

```
?- calculate_relief(11).
User Relief changed from 0.4 to 0.16666666666666666
true.
```

Figura 27 – Execução na consola do predicado calculate_relief/2

Base de conhecimento depois de calcular o novo valor:

```
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.276,0.3,0.16666666666666666,0.5,0.1,19).
no(12,beatriz,[natureza,musica,carros,porto,moda],0.67,0.4,0.6,0.5,0.3,90).
no(1,ana,[natureza,pintura,musica,sw,porto],0.999996875,0.7,0.3333333333333333,0.5652173913043478,0.6,190).
no(200,preto,[natureza,bolachas,moda,musica,sw,coimbra],0.565,0.5,0,0.4166666666666667,0.5,26).
no(21,eduardo,[natureza,cinema,teatro,carros,coimbra],0.78975,0.9,0.9,0.9,0.9,-90).
no(13,carlos,[natureza,musica,sw,futebol,coimbra],0.1,0.2,0.3,0.1,0,10).
no(14,daniel,[natureza,cinema,jogos,sw,moda],0.8,0.8,0.8,0.6,0.8,1).
no(22,isabel,[natureza,musica,porto,lisboa,cinema],0.8,0.8,0.8,0.8,0.8,100).
no(23,jose,[natureza,pintura,sw,musica,carros,lisboa],0.7,0.7,0.7,0.7,0.8,80).
no(24,luisa,[natureza,cinema,jogos,moda,porto],0.6,0.6,0.6,0.6,0.6,112).
no(31,maria,[natureza,pintura,musica,moda,porto],0.5,0.5,0.5,0.5,0.5,312).
no(32,anabela,[natureza,cinema,musica,tecnologia,porto],0.4,0.4,0.4,0.4,0.4,51).
no(33,andre,[natureza,carros,futebol,coimbra],0.3,0.3,0.3,0.3,0.3,42).
no(34,catia,[natureza,musica,cinema,lisboa,moda],0.2,0.2,0.2,0.2,0.2,45).
no(41,garcia,[natureza,teatro,tecnologia,futebol,porto,bolachas],0.5,0.5,0.5,0.5,0.5,35).
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.5,0.5,0.5,0.5,78).
no(43,fraga,[natureza,teatro,carros,porto,bolachas],0.5,0.5,0.5,0.5,0.5,73).
no(44,isaura,[natureza,moda,tecnologia,cinema],0.2,0.3,0.4,0.4,0.3,150).
no(51,rodolfo,[natureza,musica,sw],0.2,0.3,0.4,0.5,0.4,137).
no(61,rita,[moda,tecnologia,cinema],0.2,0.3,0.4,0.5,0.5,-5).
```

Figura 28 - Base de conhecimentos depois da execução do predicado calculate_relief/2

Como se pode verificar, o no 11, que foi o selecionado, passou para primeiro na base de conhecimento, isto acontece porque foi o no a ser alterado, podemos também verificar que o valor do seu alívio passou de 0.4 para 0.16(6), como indicado na execução do programa.

Cálculo do orgulho (Pride - 7º parâmetro do predicado no):

```
calculate_pride(UserId):-
  friend_list_of_user(UserId, FriendList),
  retract(no(UserId,A,UserTags,C,D,E,InitialPride,F,G)),
  tags_in_common(UserTags,FriendList,CommonTags,TotalTags),
  flatten(CommonTags,TagsComuns),flatten(TotalTags,TotalDeTags),
  length(TagsComuns,X),
  length(TotalDeTags,Y),
  % Pride is InitialPride + (1 - InitialPride) * (X/Y),
  Pride is (X/ max(Y, 1)),
  asserta(no(UserId,A,UserTags,C,D,E,Pride,F,G)),
  save(),
    write('User Pride changed from '), write(InitialPride), write(' to '), write(Pride).
```

Figura 29 - calculate_pride/1

Como o orgulho depende da percentagem do número de tags do utilizador iguais a tags de amigos em relação ao número total de tags dos amigos do mesmo, o predicado primeiro vai buscar a lista de amigos do utilizador selecionado, pela mesma lógica do predicado que calcula o alívio, de seguida faz retract ao no do utilizador selecionado, depois, através do predicado tags_in_common, obtém a lista de tags em comum entre o utilizador selecionado e as tags dos seus amigos:

```
tags_in_common(_,[],[],[]):- !.
tags_in_common(UserTags,[HeadUser|FriendList],[ListaComum|CommonTags],[Tags|TotalTags]):-
  no(HeadUser,_,Tags,_,_,_,_,_,_),
  intersecao(UserTags,Tags,ListaComum),
  tags_in_common(UserTags,FriendList,CommonTags,TotalTags).
```

Figura 30 - tags_in_common/4

Este predicado faz a interseção entre a lista de tags do user selecionado e a lista de tags de todos os seus amigos.

Voltando ao predicado principal, de seguida é feito flatten às listas de tags em comum com os amigos e total de tags dos amigos, e depois é calculado o novo valor do orgulho, sendo então feito o assert do no com os novos valores, seguido do predicado save.

Exemplo:

Base de conhecimento inicial:

```
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.276,0.3,0.16666666666666666,0.5,0.1,19).
no(12,beatriz,[natureza,musica,carros,porto,moda],0.67,0.4,0.6,0.5,0.3,90).
no(1,ana,[natureza,pintura,musica,sw,porto],0.999996875,0.7,0.3333333333333333,0.5652173913043478,0.6,190).
no(200,preto,[natureza,bolachas,moda,musica,sw,coimbra],0.565,0.5,0,0.4166666666666667,0.5,26).
no(21,eduardo,[natureza,cinema,teatro,carros,coimbra],0.78975,0.9,0.9,0.9,0.9,-90).
no(13,carlos,[natureza,musica,sw,futebol,coimbra],0.1,0.2,0.3,0.1,0,10).
no(14,daniel,[natureza,cinema,jogos,sw,moda],0.8,0.8,0.8,0.6,0.8,1).
no(22,isabel,[natureza,musica,porto,lisboa,cinema],0.8,0.8,0.8,0.8,0.8,100).
no(23,jose,[natureza,pintura,sw,musica,carros,lisboa],0.7,0.7,0.7,0.7,0.8,80).
no(24,luisa,[natureza,cinema,jogos,moda,porto],0.6,0.6,0.6,0.6,0.6,112).
no(31,maria,[natureza,pintura,musica,moda,porto],0.5,0.5,0.5,0.5,0.5,312).
no(32,anabela,[natureza,cinema,musica,tecnologia,porto],0.4,0.4,0.4,0.4,0.4,51).
no(33,andre,[natureza,carros,futebol,coimbra],0.3,0.3,0.3,0.3,0.3,42).
no(34,catia,[natureza,musica,cinema,lisboa,moda],0.2,0.2,0.2,0.2,0.2,45).
no(41,garcia,[natureza,teatro,tecnologia,futebol,porto,bolachas],0.5,0.5,0.5,0.5,0.5,35).
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.5,0.5,0.5,0.5,78).
no(43,fraga,[natureza,teatro,carros,porto,bolachas],0.5,0.5,0.5,0.5,0.5,73).
no(44,isaura,[natureza,moda,tecnologia,cinema],0.2,0.3,0.4,0.4,0.3,150).
no(51,rodolfo,[natureza,musica,sw],0.2,0.3,0.4,0.5,0.4,137).
no(61,rita,[moda,tecnologia,cinema],0.2,0.3,0.4,0.5,0.5,-5).
```

Figura 31 - Base de conhecimentos antes da execução do predicado calculate_pride/1

Cálculo do novo valor de orgulho para o utilizador 13:

```
?- calculate_pride(13).
User Pride changed from 0.1 to 0.4230769230769231
true.
```

Figura 32 - Execução na consola do predicado calculate_pride/1

Base de conhecimento depois de calcular o novo valor:

```
no(13,carlos,[natureza,musica,sw,futebol,coimbra],0.1,0.2,0.3,0.4230769230769231,0,10).
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.276,0.3,0.16666666666666666,0.5,0.1,19).
no(12,beatriz,[natureza,musica,carros,porto,moda],0.67,0.4,0.6,0.5,0.3,90).
no(1,ana,[natureza,pintura,musica,sw,porto],0.999996875,0.7,0.3333333333333333,0.5652173913043478,0.6,190).
no(200,preto,[natureza,bolachas,moda,musica,sw,coimbra],0.565,0.5,0,0.4166666666666667,0.5,26).
no(21,eduardo,[natureza,cinema,teatro,carros,coimbra],0.78975,0.9,0.9,0.9,0.9,-90).
no(14,daniel,[natureza,cinema,jogos,sw,moda],0.8,0.8,0.8,0.6,0.8,1).
no(22,isabel,[natureza,musica,porto,lisboa,cinema],0.8,0.8,0.8,0.8,0.8,100).
no(23,jose,[natureza,pintura,sw,musica,carros,lisboa],0.7,0.7,0.7,0.7,0.8,80).
no(24,luisa,[natureza,cinema,jogos,moda,porto],0.6,0.6,0.6,0.6,0.6,112).
no(31,maria,[natureza,pintura,musica,moda,porto],0.5,0.5,0.5,0.5,0.5,312).
no(32,anabela,[natureza,cinema,musica,tecnologia,porto],0.4,0.4,0.4,0.4,0.4,51).
no(33,andre,[natureza,carros,futebol,coimbra],0.3,0.3,0.3,0.3,0.3,42).
no(34,catia,[natureza,musica,cinema,lisboa,moda],0.2,0.2,0.2,0.2,0.2,45).
no(41,garcia,[natureza,teatro,tecnologia,futebol,porto,bolachas],0.5,0.5,0.5,0.5,0.5,35).
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.5,0.5,0.5,0.5,78).
no(43,fraga,[natureza,teatro,carros,porto,bolachas],0.5,0.5,0.5,0.5,0.5,73).
no(44,isaura,[natureza,moda,tecnologia,cinema],0.2,0.3,0.4,0.4,0.3,150).
no(51,rodolfo,[natureza,musica,sw],0.2,0.3,0.4,0.5,0.4,137).
no(61,rita,[moda,tecnologia,cinema],0.2,0.3,0.4,0.5,0.5,-5).
```

Figura 33 - Base de conhecimentos depois da execução do predicado calculate_pride/1

Como se pode verificar, o no 13, que foi o selecionado, passou para primeiro na base de conhecimento, isto acontece porque foi o no a ser alterado, podemos também verificar que o valor do seu orgulho passou de 0.1 para 0.423, como indicado na execução do programa.

Cálculo da gratidão (Gratitude - 8º parâmetro do predicado no):

```
calculate_gratitude(UserId):-
  users_in_groups_of_user(UserId,UsersList),
  flatten(UsersList,GroupUsers),
  remove_item_from_list(UserId,GroupUsers,GroupUsersWithoutSelf),
  retract(no(UserId,A,UserTags,C,D,E,F,InitialGratitude,G)),
  tags_in_common(UserTags,GroupUsersWithoutSelf,CommonTags,TotalTags),
  flatten(CommonTags,TagsComuns),flatten(TotalTags,TotalDeTags),
  length(TagsComuns,X),
  length(TotalDeTags,Y),!,
  % Gratitude is InitialGratitude + (1 - InitialGratitude) * (X/Y),
  Gratitude is (X/ max(Y, 1)),
  asserta(no(UserId,A,UserTags,C,D,E,F,Gratitude,G)),
  save(),
    write('User Gratitude changed from '), write(InitialGratitude), write(' to '), write(Gratitude).
```

Figura 34 - calculate_gratitude/1

Como a gratidão depende da percentagem do número de tags do utilizador em grupos do mesmo em relação ao número total de tags nos grupos do mesmo, o predicado assemelha-se muito ao predicado que calcula o alívio, visto que a lógica é a mesma, sendo que a única diferença é a mudança de amigos para tags.

É obtida a lista de utilizadores em grupos do utilizador selecionado, exceto o próprio, através dos predicados users_in_groups_of_user, flatten e remove_item_from_list. De seguida é obtida a lista de tags e o valor inicial da gratidão do user selecionado, com o retract, e é utilizado o predicado tags_in_common para obter a lista de tags que o utilizador selecionado tem em comum com a lista total de tags de todos os seus grupos, utilizando a mesma lógica explicada no predicado de calcular o orgulho.

Finalmente, é calculado o novo valor de gratidão, e é feito o assert do no com o novo valor de gratidão, seguido do predicado save.

Exemplo:

Base de conhecimento inicial:

```
no(13,carlos,[natureza,musica,sw,futebol,coimbra],0.1,0.2,0.3,0.4230769230769231,0,10).
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.276,0.3,0.16666666666666666,0.5,0.1,19).
no(12,beatriz,[natureza,musica,carros,porto,moda],0.67,0.4,0.6,0.5,0.3,90).
no(1,ana,[natureza,pintura,musica,sw,porto],0.999996875,0.7,0.3333333333333333,0.5652173913043478,0.6,190).
no(200,preto,[natureza,bolachas,moda,musica,sw,coimbra],0.565,0.5,0,0.4166666666666667,0.5,26).
no(21,eduardo,[natureza,cinema,teatro,carros,coimbra],0.78975,0.9,0.9,0.9,0.9,-90).
no(14,daniel,[natureza,cinema,jogos,sw,moda],0.8,0.8,0.8,0.6,0.8,1).
no(22,isabel,[natureza,musica,porto,lisboa,cinema],0.8,0.8,0.8,0.8,0.8,100).
no(23,jose,[natureza,pintura,sw,musica,carros,lisboa],0.7,0.7,0.7,0.7,0.8,80).
no(24,luisa,[natureza,cinema,jogos,moda,porto],0.6,0.6,0.6,0.6,0.6,112).
no(31,maria,[natureza,pintura,musica,moda,porto],0.5,0.5,0.5,0.5,0.5,312).
no(32,anabela,[natureza,cinema,musica,tecnologia,porto],0.4,0.4,0.4,0.4,0.4,51).
no(33,andre,[natureza,carros,futebol,coimbra],0.3,0.3,0.3,0.3,0.3,42).
no(34,catia,[natureza,musica,cinema,lisboa,moda],0.2,0.2,0.2,0.2,0.2,45).
no(41,garcia,[natureza,teatro,tecnologia,futebol,porto,bolachas],0.5,0.5,0.5,0.5,0.5,35).
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.5,0.5,0.5,0.5,78).
no(43,fraga,[natureza,teatro,carros,porto,bolachas],0.5,0.5,0.5,0.5,0.5,73).
no(44,isaura,[natureza,moda,tecnologia,cinema],0.2,0.3,0.4,0.4,0.3,150).
no(51,rodolfo,[natureza,musica,sw],0.2,0.3,0.4,0.5,0.4,137).
no(61,rita,[moda,tecnologia,cinema],0.2,0.3,0.4,0.5,0.5,-5).
```

Figura 35 - Base de conhecimentos antes da execução do predicado calculate_gratitude/1

Cálculo do novo valor de gratidão para o utilizador 14:

```
?- calculate_gratitude(14).
User Gratitude changed from 0.8 to 0.42857142857142855
true.
```

Figura 36 - Execução na consola do predicado calculate_gratitude/1

Base de conhecimento depois de calcular o novo valor:

```
no(14,daniel,[natureza,cinema,jogos,sw,moda],0.8,0.8,0.8,0.6,0.42857142857142855,1).
no(13,carlos,[natureza,musica,sw,futebol,coimbra],0.1,0.2,0.3,0.4230769230769231,0,10).
no(11,antonio,[natureza,pintura,carros,futebol,lisboa],0.276,0.3,0.16666666666666666,0.5,0.1,19).
no(12,beatriz,[natureza,musica,carros,porto,moda],0.67,0.4,0.6,0.5,0.3,90).
no(1,ana,[natureza,pintura,musica,sw,porto],0.999996875,0.7,0.3333333333333333,0.5652173913043478,0.6,190).
no(200,preto,[natureza,bolachas,moda,musica,sw,coimbra],0.565,0.5,0,0.4166666666666667,0.5,26).
no(21,eduardo,[natureza,cinema,teatro,carros,coimbra],0.78975,0.9,0.9,0.9,0.9,-90).
no(22,isabel,[natureza,musica,porto,lisboa,cinema],0.8,0.8,0.8,0.8,0.8,100).
no(23,jose,[natureza,pintura,sw,musica,carros,lisboa],0.7,0.7,0.7,0.7,0.8,80).
no(24,luisa,[natureza,cinema,jogos,moda,porto],0.6,0.6,0.6,0.6,0.6,112).
no(31,maria,[natureza,pintura,musica,moda,porto],0.5,0.5,0.5,0.5,0.5,312).
no(32,anabela,[natureza,cinema,musica,tecnologia,porto],0.4,0.4,0.4,0.4,0.4,51).
no(33,andre,[natureza,carros,futebol,coimbra],0.3,0.3,0.3,0.3,0.3,42).
no(34,catia,[natureza,musica,cinema,lisboa,moda],0.2,0.2,0.2,0.2,0.2,45).
no(41,garcia,[natureza,teatro,tecnologia,futebol,porto,bolachas],0.5,0.5,0.5,0.5,0.5,35).
no(42,patricia,[natureza,futebol,sw,jogos,porto,bolachas],0.5,0.5,0.5,0.5,0.5,78).
no(43,fraga,[natureza,teatro,carros,porto,bolachas],0.5,0.5,0.5,0.5,0.5,73).
no(44,isaura,[natureza,moda,tecnologia,cinema],0.2,0.3,0.4,0.4,0.3,150).
no(51,rodolfo,[natureza,musica,sw],0.2,0.3,0.4,0.5,0.4,137).
no(61,rita,[moda,tecnologia,cinema],0.2,0.3,0.4,0.5,0.5,-5).
```

Figura 37 - Base de conhecimentos depois da execução do predicado calculate_gratitude/1

Como se pode verificar, o no 14, que foi o selecionado, passou para primeiro na base de conhecimento, isto acontece porque foi o no a ser alterado, podemos também verificar que o valor da sua gratidão passou de 0.8 para 0.429, como indicado na execução do programa.

Estado da Arte

Machine Learning application in the problem of dealing with emotional aspects in social networks

Patrícia Barbosa, Pedro Fraga, Pedro Garcia, Pedro Preto
Instituto Superior de Engenharia do Porto

I. Introduction

Nowadays, people have become increasingly accustomed to expressing their opinions online. The rapid growth of social media platforms provides rich multimedia data in large quantities for various research opportunities such as sentiment analysis. Microblog data contains a large amount of valuable sentiment information, which is not only of interest for commercial use, but also for psychology, cognitive linguistics, or political science [1].

Sentiment analysis is a popular topic in the study and better understanding of social networks and represents one of the most fundamental aspects of social network analysis [2]. Determining user behaviour in each of the emerging communities is a vivid concept for analysing exactly how users come together to form social communities [2].

II. Machine Learning And Emotional Aspects On Social Media

Machine learning is a type of artificial intelligence that allows software applications predict outcomes more effectively without being directly programmed to do so [3]. To do that, the algorithms, using historical data as feedback, simulate new output values[3].

Machine learning has been used in text-based classification and object recognition with great success in a variety of applications, including sentiment and affect analysis [4].

One of the basics behaviors of Artificial intelligence is machine learning, and is one of the most important, since cannot exist intelligence without learning [5]." Machine learning algorithms were from the very beginning designed and used to analyse medical datasets" [5].

Since the firsts electronic computers were invented in the 1950s, the algorithms developed that enabled modelling and analysing large sets of data, have been in constant evolution and development, besides that, along time machine learning branched in different in several work areas[5] .

In the nowadays, machine learning is an imperative tool for intelligent data analysis, since is one of the best tools to collect data and process it [5].

Emotions are described as intense feelings directed toward something or someone in response to internal or external events that have special meaning for the individual [6]. The Internet today has become an important medium through which people express their emotions, feelings, and opinions. Events, news, or activities are shared, discussed, posted and commented on by millions of people on social media [6]. Capturing these emotions in texts, especially those posted or shared on social media, can be a source of valuable information that can be used to study how different people react to different situations and events [6]. Business analysts use this type of information to determine people's feelings and opinions about their products [6].

Emotions play an essential role in conversation because they interact with the context of the text to express meanings for the conversation [7]. Recognizing emotions from texts has become an attractive task in recent years but, with the advances in machine learning techniques and hardware to support the machine learning process, has constituted a closer reality [7].

III. Machine Learning Application

"Researchers from Skoltech, INRIA and the RIKEN Advanced Intelligence Project have considered several state-of-the-art machine learning algorithms for the challenging tasks of determining the mental workload and affective states of a human brain. Their software can help design smarter brain-computer interfaces for applications in medicine and beyond "[8].

By using a brain-computer interface, or BCI, that represents a connection between a human brain and a machine, it becomes possible to monitoring the mental state and emotions of a user, and consequently categorize them, these are known by "passive BCIs"[8]. "Brain signals in a BCI are usually measured by electroencephalography, a typically noninvasive method of recording electrical activity of the brain"[8] .

Some examples of machine learning algorithms emotional states oriented are: Riemannian geometry-based classifiers (RGC) and convolutional neural networks (CNN). To discover how this algorithm, work with emotional states predictions, the scientists "ran a competition of sorts among seven algorithms, two of which the scientists designed themselves by further improving well-performing Riemannian methods. The algorithms were tested in two studies, one with a typical arrangement for BCIs where algorithms were trained on data from a specific subject and later tested on that same subject, and one that was subject-independent" [8]." The scientists found, for instance, that an artificial deep neural network outperformed all its competitors quite significantly in the workload estimation task but did poorly in emotion

classification. And the two modified Riemannian algorithms did quite well in both tasks" [8].

IV. Related Works

In the recent past, with the exponential growth of social media, has been verified an interest on sentiment analysis. This promoted a lot of research related with machine learning algorithms to processing the emotional aspects on social media.

The writers of [3] focus their research on a optimization of machine learning programming algorithms for sentiment analysis in social media. In this work, it is assumed that in sentiment analysis the feature selection phase is crucial, since there is not yet a best way to solve all optimization problems [3]. It was used a hybrid approach combining hybrid algorithms [(GWO and (PSO)] and to reduce the search space, filter out the features selection [3]. This combination, GWOPS, is used for training neural network (NN) classifiers to select the best features through data collected from Twitter on different topics [3]. The algorithm was compared with other three well-known optimization algorithms aiming to evaluate the quality and effectiveness of GWOPS[3].

Another paper proposed a machine learning method that considers Internet slang and emoticons for sentiment analysis of the most popular Chinese social media platform [1]. Briefly, they collected expressions as a slang lexicon and converted the 109 emoticons of the platform into text features to create a Chinese emoticon lexicon [1]. Then, both lexicons were used with different machine learning approaches to test the ability of detecting humorous posts [1]. The results of this experiment show that the proposed method can significantly improve expression recognition performance [1].

The researchers of the other paper propose a method for classifying texts into six different emotion categories: Joy, Sadness, Fear, Anger, Surprise, and Disgust [6]. In the model, two different approaches are used and combined to effectively extract these emotions from the text [6]. The first approach is based on natural language processing, the other on machine learning classification algorithms [6]. The results of these experiments show that the model provides significant accuracy in classifying tweets from Twitter [6].

Another study, focusing more on negative emotions transmitted on social media, believes that there is growing evidence that the Internet and social media provide clues to people's emotional distress, such as messages indicating emotional distress, or even leaving suicide messages on the Internet [4]. They present a design of a system called KAREN that identifies people who blog about their emotional distress in Chinese through a combination of machine learning and rule-based classification with rules determined by experts [4]. The results show that the proposed system achieved better classification performance than benchmark methods, and experts found the system more useful and effective for identifying bloggers with emotional distress than benchmark approaches [4].

In the last paper analysed, several popular machine learning techniques for recognising emotions from a conversation in social media were investigated [7]. The algorithms proposed range from traditional machine learning to deep learning techniques [7]. The research contributes to extensive studies of several machine learning algorithms [7].

V. Conclusion

From this study and analysis, we can conclude that machine learning techniques for analyzing emotions are widely used in social networks. From commercial purposes, such as knowing the opinion of consumers regarding products, to health purposes, such as detecting negative emotional states and signs of mental problems, there are many works dealing with this topic.

It is a growing technology, and for this reason there is a lot of work in this area, but there are still some points that need to be improved, such as recognizing specific feelings instead of negative feelings versus positive feelings.

VI. References

[1] D. Li, R. Rzepka, M. Ptaszynski, and K. Araki, "A Novel Machine Learning-based Sentiment Analysis Method for Chinese Social Media Considering Chinese Slang Lexicon and Emoticons." [Online]. Available: https://www.quintly.com/blog/instagram-emoji-study

[2] A. Kanavos, I. Perikos, I. Hatzilygeroudis, and A. Tsakalidis, "Emotional community detection in social networks," *Computers & Electrical Engineering*, vol. 65, pp. 449–460, Jan. 2018, doi: 10.1016/J.COMPELECENG.2017.09.011.

[3] M. S. F. Alharbi and E.-S. M. El-kenawy, "Optimize Machine Learning Programming Algorithms for Sentiment Analysis in Social Media," *International Journal of Computer Applications*, vol. 174, no. 25, pp. 38–43, Mar. 2021, doi: 10.5120/ijca2021921169.

[4] M. Chau, T. M. H. Li, P. W. C. Wong, J. J. Xu, P. S. F. Yip, and H. Chen, "Finding people with emotional distress in online social media: A design combining machine learning and rule-BASED classification," *MIS Quarterly: Management Information Systems*, vol. 44, no. 2, pp. 933–956, Jun. 2020, doi: 10.25300/MISQ/2020/14110.

[5] I. Kononenko, "Machine learning for medical diagnosis: history, state of the art and perspective," *Artificial Intelligence in Medicine*, vol. 23, no. 1, pp. 89–109, Aug. 2001, doi: 10.1016/S0933-3657(01)00077-X.

[6] B. Gaind, V. Syal, and S. Padgalwar, "Emotion Detection and Analysis on Social Media," Jan. 2019, [Online]. Available: http://arxiv.org/abs/1901.08458

[7] A. Chowanda, R. Sutoyo, Meiliana, and S. Tanachutiwat, "Exploring Text-based Emotions

Recognition Machine Learning Techniques on Social Media Conversation," in *Procedia Computer Science*, 2021, vol. 179, pp. 821–828. doi: 10.1016/j.procs.2021.01.099.

[8] "Skoltech | Machine learning, meet human emotions: How to help a computer monitor your mental state." https://www.skoltech.ru/en/2020/08/machine-learning-meet-human-emotions-how-to-help-a-computer-monitor-your-mental-state/ (accessed Jan. 23, 2022).

Conclusões

Resumindo, a aprendizagem automática é uma técnica que permite analisar estados emocionais nas redes sociais. No contexto deste projeto, para cada interação que o utilizador tenha, os seus estados emocionais são manipulados e os seus valores atualizados.

Os valores atualizados dos estados emocionais permitem filtrar os caminhos, através do Best First (adaptado ao uso de emoções), de forma a que apenas caminhos para users com valores acima de 0.5, inclusive, sejam considerados.

Para outra vertente do projeto adaptou-se o predicado do primeiro sprint que devolvia os users com X tags em comum de forma a considerar, para além das tags em comum, um conjunto de tags obrigatórias, e devolver os de users que partilham das mesmas.