23-01-2022

Relatório ALGAV

Sprint D

Grupo:

- Helder Serralva 1181180
- Diogo Silva 1181178
- Diogo Ribeiro 1190508
- Ricardo Cardoso 1171388

Turma 3DE

Índice

Consideração de estados emocionais para encontrar caminhos

O requisito consiste em considerar estados emocionais para encontra caminhos (por exemplo, não englobar nenhum utilizador que apresente estados de angústia, medo, deceção, remorsos e raiva superiores a 0,5 numa escala entre 0 e 1). Esta explicação não nos pareceu muito concreta, e ficaram algumas dúvidas sobre aquilo que realmente era pretendido. A nossa abordagem foi ter em consideração uma lista de emoções proibidas, ou seja, todas as emoções que estivessem nessa lista não poderiam ter um valor superior a 0.5, valor definido como neutro. Como é obvio, se esta lista estiver vazia significa que não existem restrições de emoções.

Para isto, foi desenvolvida um predicado que recebe no primeiro parâmetro uma lista com todas as emoções e os respetivos valores e no segundo parâmetro a lista de emoções proibidas. Este predicado percorre todas as emoções da primeira lista e para cada uma verifica se essa emoção existe na segunda lista. No caso de não existir passa á próxima, no caso de existir verifica se o valor da mesma é igual ou inferior a 0.5.

```
verificarEmocoes(_, []):-!.
verificarEmocoes([],_):-!.

verificarEmocoes([(Emocao,_)|Lem], Emocoes):-
                \+ member(Emocao, Emocoes),
                verificarEmocoes(Lem,Emocoes),!.

verificarEmocoes([(Emocao, Valor)|Lem], Emocoes):-
                member(Emocao, Emocoes),
                Valor =< 0.5,
                verificarEmocoes(Lem, Emocoes).
```

Depois de implementado, bastava integrá-lo nos algoritmos de pesquisa desenvolvidos nos sprints anteriores. A abordagem foi a mesma para os 3, foi adicionado um parâmetro extra no final que permite identificar estas emoções proibidas. Sempre que é encontrado uma nova ligação válida, o algoritmo procura a lista de emoções do novo nó e chama este método acima descrito, passando essas emoções no primeiro parâmetro. De notar, que esta chamada só acontece caso o nó encontrado não seja o nó destino, uma vez que se pretende apenas restringir as emoções no caminho entre os nós, ou seja, as emoções do nó origem e do nó destino não são tidas em conta. Podemos ver a integração do predicado nos algoritmos DFS, A* e BestFirst abaixo.

```
dfs(Orig,Dest,Cam, FCam, MForca, MaxLigacoes, EmocoesProibidas):-
                dfs2(Orig,Dest,[Orig],0,101, Cam, FCam, MForca, MaxLigacoes, EmocoesProibidas).

dfs2(Dest,Dest,LA,FA,MF, Cam, FA, MF,_, _):-!,reverse(LA,Cam).
dfs2(Act,Dest,LA,FA,MF, Cam, FCam, MForca, ML,EmocoesProibidas):-
                length(LA, Tam),
                Tam =< ML,
                (ligacao(Act,NX,FL1,FL2,_,_);ligacao(NX,Act,FL2,FL1,_,_)),
                \+ member(NX,LA),
                (NX == Dest;(no(NX,_,Emocao),verificarEmocoes(Emocao,EmocoesProibidas))),
                menor_forca(FL1,FL2,MF,Menor),
                FT is FA + FL1,
                dfs2(NX,Dest,[NX|LA],FT,Menor, Cam, FCam, MForca, ML, EmocoesProibidas).
```

```
aStar2(Dest,[(_,Ca,LA)|Outros],MaxLigacoes,Cam,Custo, Forcas, EmocoesProibidas):-
    length(LA, Tam),
    Tam =< MaxLigacoes,
    LA=[Act|_],
    findall((CEX,CaX,[X|LA]),
            (Dest\==Act,(ligacao(Act,X,FLX,_,_,_);ligacao(X,Act,_,FLX,_,_)),\+ member(X,LA),
             (X == Dest;(no(X,_,Emocao),verificarEmocoes(Emocao,EmocoesProibidas))),
             append([X], LA, T),
             removerForcasUsadas(T, Forcas, ForcasRestantes),
             NiveisRestantes is MaxLigacoes - Tam,
             CaX is FLX + Ca, estimativa(ForcasRestantes,NiveisRestantes,EstX),
             CEX is CaX +EstX),Novos),

    append(Outros,Novos,Todos),
    sort(0, @>=, Todos,TodosOrd),
    aStar2(Dest,TodosOrd,MaxLigacoes,Cam,Custo, Forcas,EmocoesProibidas).


bestfs12(Dest,LLA,MaxLigacoes,Cam,Custo,EmocoesProibidas):-
       member1(LA,LLA,LLA1),
       length(LA, Tam),
       Tam =< MaxLigacoes,
       LA=[Act|_],
       ((Act==Dest,!,bestfs12(Dest,[LA|LLA1],MaxLigacoes,Cam,Custo,EmocoesProibidas))
        ;
        (
         findall((CX,[X|LA]),((ligacao(Act,X,CX,_,_,_);ligacao(X,Act,_,CX,_,_)),
         \+member(X,LA),
         (X == Dest;(no(X,_,Emocao),verificarEmocoes(Emocao,EmocoesProibidas))))
         ,Novos),
         Novos\==[],!,
         sort(0,@>=,Novos,NovosOrd),
         retira_custos(NovosOrd,NovosOrd1),
         append(NovosOrd1,LLA1,LLA2),
         bestfs12(Dest,LLA2,MaxLigacoes,Cam,Custo,EmocoesProibidas)
        )).
```

Exemplos:

- Usar o DFS para encontrar o caminho mais forte do 1 para o 8, sem qualquer restrição de emoções. O caminho gerado é 1,2,5,8. Na nossa rede o utilizador 5 tem por exemplo a emoção “Aliviado” a 0.6, ou seja, se definirmos essa emoção como sendo uma emoção proibida o caminho gerado já será outro. Se ainda tivermos em conta mais emoções como por exemplo a do utilizador 6 que é “Desapontado”, esse nó também não deve constar no resultado final.

```
?- maisForte(1,8,10,Caminho,_,Custo,_,[]).
Caminho = [1, 2, 5, 8],
Custo = 22.

?- maisForte(1,8,10,Caminho,_,Custo,_,["Aliviado"]).
Caminho = [1, 4, 3, 6, 8],
Custo = 9.

?- maisForte(1,8,10,Caminho,_,Custo,_,["Aliviado","Desapontado"]).
Caminho = [1, 4, 3, 7, 8],
Custo = 8.
```

- Usar o A* para encontra um caminho de 1 a 8 tendo em conta as mesmas restrições.

```
?- aStar(1,8,10,Caminho,Custo,[]).
Caminho = [1, 2, 5, 8],
Custo = 22.

?- aStar(1,8,10,Caminho,Custo,["Aliviado"]).
Caminho = [1, 4, 3, 6, 8],
Custo = 9.

?- aStar(1,8,10,Caminho,Custo,["Aliviado", "Desapontado"]).
Caminho = [1, 4, 3, 7, 8],
Custo = 8.
```

- Usar o BestFirst para encontrar o mesmo caminho nas mesmas condições.

```
?- bestfs(1,8,10,Caminho,Custo,[]).
Caminho = [1, 2, 5, 8],
Custo = 22.

?- bestfs(1,8,10,Caminho,Custo,["Aliviado"]).
Caminho = [1, 4, 3, 6, 8],
Custo = 9.

?- bestfs(1,8,10,Caminho,Custo,["Aliviado", "Desapontado"]).
Caminho = [1, 4, 3, 7, 8],
Custo = 8.
```

Sugestões de Grupos

Pretendia-se que fosse desenvolvido também um método que sugerisse o maior grupo para um dado utilizador (conhecido) desde que tenha um mínimo de N utilizadores e T tags comuns, sendo que devia ser possível definir algumas tags como sendo obrigatórias.

Para esta requisito, foi usado o predicado implementado no sprint B "x_tags_em_comum". O nosso predicado devolve uma combinação de tags e a respetiva lista de utilizadores que as partilham. Assim, fazendo uso do *backtracking* recorremos ao "findall" para obter todas as combinações possíveis com T tags em comum. Depois de obter uma solução, começamos por verificar se o utilizador que pediu a sugestão está inserido na lista de utilizadores dessa solução (1). Uma vez que o número de utilizadores de um grupo não deve considerar o próprio utilizador, removemos esse utilizador da lista (2). De seguida, verificamos se o número mínimo de utilizadores se verifica (3). No fim, é verificado se as tags obrigatórias estão presentes também (4). Todas as soluções que verificarem estas 4 condições dão origem aos grupos sugeridos para o utilizador.

Optamos por definir também uma condição no início que impede que o número de tags obrigatórias seja maior que o número de tags definidas.

```
sugerirGrupos(Utilizador, NumeroTags,TagsObrigatorias,NumeroUtilizadores, Grupos):-
                length(TagsObrigatorias, TamanhoObrigatorias),
                TamanhoObrigatorias =< NumeroTags,
                findall((USU,T),
                        (   x_tags_em_comum(NumeroTags,T,U),
                         (1) member(Utilizador,U),
                         (2) delete(U, Utilizador, USU),
                            length(USU, Tam),
                         (3) Tam >= NumeroUtilizadores,
                         (4) temTagsObrigatorias(TagsObrigatorias,T)
                        ),
                        Grupos).


temTagsObrigatorias([],_).

temTagsObrigatorias([Obrigatoria|Obrigatorias], Tags):-
                 temTag(Obrigatoria,Tags),
                 temTagsObrigatorias(Obrigatorias, Tags).
```

Podem ser analisados alguns exemplos abaixo.

- O utilizador 7 procura sugestões de grupos com 3 tags em comum, com pelo menos 2 utilizadores e sem nenhuma tag obrigatória. Foram devolvidos dois grupos, o primeiro com o utilizador 3 e 1, onde "natureza", "musica" e "porto" são as tags em comum e o segundo com os utilizadores 9 e 8 que partilham as tags "lisboa", "natureza" e "musica".

```
?- sugerirGrupos(7,3,[],2,Grupos).
Grupos = [([3, 1], [natureza, musica, porto]),  ([9, 8], [lisboa, natureza, musica])].
```

- Vamos agora fazer a mesma pesquisa, mas definindo “natureza” e “musica” como tags obrigatórias, como deve ser esperado, o resultado é o mesmo uma vez que ambas as soluções possuem as duas tags.

```
?- sugerirGrupos(7,3,["natureza", "musica"],2,Grupos).
Grupos = [([3, 1], [natureza, musica, porto]), ([9, 8], [lisboa, natureza, musica])].
```

- Se agora definirmos “porto” como sendo a única tag obrigatória, só devemos receber o grupo com os utilizadores 3 e 1.

```
?- sugerirGrupos(7,3,["porto"],2,Grupos).
Grupos = [([3, 1], [natureza, musica, porto])].
```

Calcular os novos valores dos estados emocionais

Aqui foi pedido que fosse desenvolvido métodos que permitissem atualizar determinadas emoções em função de determinados objetivos. Por exemplo, as emoções Alegria e Angústia dependendo da diferença de *likes* e *dislikes* e as emoções Esperança, Medo, Alívio e Deceção baseando-se no resultado duma sugestão de grupos.

A primeira coisa feita, foi implementar as fórmulas de aumentar ou diminuir o valor de uma determinada emoção. Aqui não existe muito a explicar uma vez que foram seguidas as fórmulas apresentados nos pdfs de apoio.

```
aumentarEmocao(EmocaoT,Valor,ValorMaximo,EmocaoT1):-
                minimo(Valor,ValorMaximo, M),
                V1 is M / ValorMaximo,
                V2 is 1 - EmocaoT,
                EmocaoT1 is EmocaoT + (V2 * V1).

diminuirEmocao(EmocaoT,Valor,ValorMaximo,EmocaoT1):-
                minimo(Valor,ValorMaximo,M),
                V1 is M /ValorMaximo,
                V2 is 1 - V1,
                EmocaoT1 is EmocaoT * V2.
```

De seguida, passamos á implementação da variação que dependem da diferença de *likes e dislikes*, ou seja, a variação das emoções alegria e angústia. Estes métodos também são bastante simples, uma vez que se baseiam naquilo que é descrito nos *pdf's*. Existem 3 situações diferentes. A primeira, caso o número de *likes* seja igual ao número de *dislikes*, as duas emoções não são alteradas e o mesmo valor é retornado [1]. Temos depois o caso em que o número de *likes* é maior que o número de *dislikes* e aí deve-se aumentar a emoção Alegria e diminuir a emoção Angústia [2]. No terceiro caso, em que o número de *dislikes* é maior que o número de *likes*, as coisas invertem-se, ou seja, a alegria deve diminuir e a angústia aumentar [3]. Definimos também um valor de saturação a 200 [4].

```
atualizarAlegriaAngustia(AlegriaT,AngustiaT,Likes,Likes,AlegriaT,AngustiaT):-!. [1]
atualizarAlegriaAngustia(AlegriaT, AngustiaT,Likes,Dislikes, AlegriaT1, AngustiaT1):-  [2]
                DF is Likes - Dislikes,
                DF > 0,
                valorSaturacaoLikesDislikes(S),
                aumentarEmocao(AlegriaT,DF,S,AlegriaT1),
                diminuirEmocao(AngustiaT,DF,S,AngustiaT1),!.

atualizarAlegriaAngustia(AlegriaT, AngustiaT,Likes,Dislikes, AlegriaT1, AngustiaT1):- [3]
                DF is Likes - Dislikes,
                DF < 0,
                D is 0 - DF,
                valorSaturacaoLikesDislikes(S),
                aumentarEmocao(AngustiaT,D,S,AngustiaT1),
                diminuirEmocao(AlegriaT,D,S,AlegriaT1),!.


valorSaturacaoLikesDislikes(200). [4]
```

Em relação ao método de atualizar as outras 4 emoções, já é um pouco mais complexo. A interpretação deste problema não foi fácil, mas seguimos o exemplo enviado pelo professor Carlos. Assim começamos por calcular o valor médio do número de utilizadores desejados e dos não desejados [1]. Uma vez que o número de utilizadores que aparecem num grupo poderia ser maior que o número de utilizadores desejados/não desejados, optamos por usar o menor valor desses dois [2]. De seguida e tendo em conta as mesmas 3 situações descritas acima, aumentamos/diminuímos cada par de emoções de acordo com o seu fator. Primeiro trata-se do para Esperanca-Dececao que se focam no número de utilizadores desejados [3] e depois dos par Alivio-Medo que baseiam-se nos utilizadores não desejados [4].

```
atualizarEsperancaMedoAlivioDececao(EsperancaT,MedoT,AlivioT,DececaoT,Desejados,NaoDesejados,DesejadosResultado,NaoDesejadosResultado,EsperancaT1,MedoT1,AlivioT1,DececaoT1):-
        VMD is Desejados / 2,
        VMND is NaoDesejados / 2,                                                    [1]

        minimo(Desejados,DesejadosResultado,DR),                                     [2]
        DD is Desejados - DR,
[3]     (   (DesejadosResultado > VMD,!,aumentarEmocao(EsperancaT,DesejadosResultado,Desejados,EsperancaT1),diminuirEmocao(DececaoT,DD,Desejados,DececaoT1));
        (DesejadosResultado < VMD,!,diminuirEmocao(EsperancaT,DesejadosResultado,Desejados,EsperancaT1),aumentarEmocao(DececaoT,DD,Desejados,DececaoT1));
        (EsperancaT1 is EsperancaT,DececaoT1 is DececaoT)),

        minimo(NaoDesejados,NaoDesejadosResultado,NDR),                              [2]
        DND is NaoDesejados - NDR,
[4]     (   (NaoDesejadosResultado > VMND,!,aumentarEmocao(AlivioT,DND,NaoDesejados,AlivioT1),diminuirEmocao(MedoT,NaoDesejadosResultado,NaoDesejados,MedoT1));
        (NaoDesejadosResultado < VMND,!,diminuirEmocao(AlivioT,DND,NaoDesejados,AlivioT1),aumentarEmocao(MedoT,NaoDesejadosResultado,NaoDesejados,MedoT1));
        (MedoT1 is MedoT,AlivioT1 is AlivioT)).
```

Podemos ver alguns exemplos abaixo:

- Variação das emoções Alegria-Angústia tendo em conta a diferença de likes e dislikes. A alegria tem como valor atual 0.6, a Angústia tem o valor 0.3 e a diferença de likes é 100 (200-100. Assim temos estes novos valores:
-

```
?- atualizarAlegriaAngustia(0.6,0.3,200,100,Alegria,Angustia).
Alegria = 0.8,
Angustia = 0.15.
```

- Em relação às emoções afetadas pelos grupos sugeridos, vamos testar o exemplo dado pelo professor Carlos. O utilizador indica 3 utilizadores desejados e 2 não desejados. Dos 3 desejados só 2 foram incluídos e dos 2 não desejados só 1 é que não foi incluído. A ter em conta as emoções inicias, em que Esperança = 0.6, Medo = 0.3, Alívio = 0.1 e Deceção = 0.8. Assim temos (de reparar que o par Medo-Alivio não se alterou uma vez que o número de utilizadores não desejados incluídos no grupo foi igual ao número de utilizadores não desejados não incluídos):

```
?- atualizarEsperancaMedoAlivioDececao(0.6,0.3,0.1,0.8,3,2,2,1,Esperanca,Medo,Alivio,Dececao)
Esperanca = 0.8666666666666667,
Medo = 0.3,
Alivio = 0.1,
Dececao = 0.5333333333333334.
```

Estado da arte – Machine Learning

Problemática em causa: Aprendizagem Automática (AA) no uso de tecnologias aplicadas ao tratamento de aspetos emocionais.

De uma forma muito sucinta o ***machine learning*** é um tópico que tem ganho cada vez mais destaque nos últimos tempos. Trata-se um tipo de inteligência artificial que permite que as aplicações de software sejam bastante precisas na previsão de resultados, mesmo sem serem expressamente programadas para isso.

O ***Sentiment Analysis*** é uma ferramenta de aprendizagem automática que analisa textos/frases quanto à "polaridade", do positivo ao negativo. Ao criar ferramentas de aprendizagem automática como por exemplos de emoções em texto, as máquinas aprendem automaticamente a detetar sentimentos sem intervenção humana. (Singh, 2017)

Para simplificar, o AA permite que os computadores aprendam novas tarefas sem serem expressamente programados para realizá-las. Os modelos de análise de sentimentos podem ser treinados para ler além de meras definições, para entender coisas como contexto, sarcasmo e palavras mal aplicadas. Por exemplo:

"Interface muito fácil de usar. Um diploma de engenharia seria útil."

Fora de contexto, as palavras "muito fácil de usar" e "útil" podem ser lidas como positivas, mas este é claramente um comentário negativo. Usando a análise de sentimentos, os computadores podem processar dados de texto automaticamente e entendê-los como um humano faria.

Como iremos ver, existem diferentes métodos ou algoritmos complexos usados para "treinar" a máquina a realizar analise de sentimentos. Como tal existem aspetos positivos e negativos para ambos, contudo, a utilização simultânea de vários pode resultar em melhores resultados.

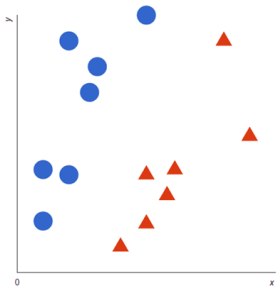
Um exemplo de uma metodo é o Naive Bayes, que se trata de um algoritmo probabilistico simples que, para uma classificação de analise de sentimentos, atribui a possibilidade de que uma palavra ou frase pode ser positiva ou negativa. (Kumar, 2021)

O cálculo da probabilidade de A, se B for verdadeiro, é igual à probabilidade de B, se A for verdadeiro, vezes a probabilidade de A ser verdadeiro, dividida pela probabilidade de B ser verdadeiro.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

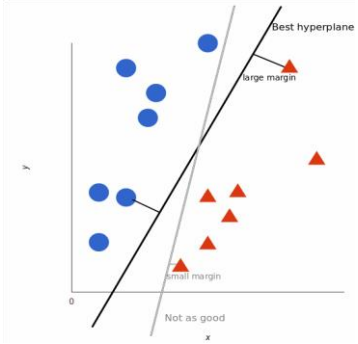
A regressão linear é um outro algoritmo usado para prever um valor *Y*, dados os recursos *X*. As relações são então colocadas ao longo do eixo *X/Y*, com uma linha reta a passar por elas para prever outras relações. Esta calcula como a entrada *X* (palavras e frases) relaciona-se com a saída *Y* (polaridade), percebendo assim se uma palavra/frase tem sentido realmente positivo ou negativo.

De forma a completar a regressão linear, temos a Support Vector Machines (SVM), que usa técnicas semelhantes à mesma, mas de forma mais avançada. Para ser mais fácil explicar, usaremos duas tags: *azul e vermelho*, com dois recursos de dados: *X* e *Y*. Assim iremos aplicar esta técnica de forma a gerar uma coordenada *X/Y como vermelha ou azul*. (Gandhi, 2018)



Assim o SVM então atribui um hiper-plano que melhor separa as tags. Em duas dimensões isso é simplesmente uma linha (como na regressão linear). Qualquer coisa de um lado da linha é *vermelha* e qualquer coisa do outro lado é *azul*. Para análise de sentimento, isso seria *positivo* e *negativo*.

Para maximizar o AA, o melhor hiper-plano é aquele com a maior distância entre cada tag:



No entanto, à medida que os conjuntos de dados se tornam mais complexos, pode não ser possível traçar uma única linha para classificar os dados, e assim torna-se necessário recorrer as três dimensões através de um eixo Z. Esta técnica devido ao facto de ser multidimensional permite uma aprendizagem automática muito mais precisa.

Neste artigo, discutimos algumas maneiras diferentes de desenvolver um modelo de análise de sentimentos. Contudo nenhum método é uma regra a seguir ao desenvolver um modelo de análise de sentimentos. Este precisa de planeamento e ajustes dos algoritmos de acordo com a declaração do problema e o conjunto de dados.

Referencias

1. Gandhi, R. (7 de Junho de 2018). *Support Vector Machine — Introduction to Machine Learning Algorithms*. Obtido de towardsdatascience.com: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

2. Kumar, S. (11 de Fevereiro de 2021). *5 Ways to develop a Sentiment Analyser in Machine Learning*. Obtido de towarsddatascience.com: https://towardsdatascience.com/5-ways-to-develop-a-sentiment-analyser-in-machine-learning-e8352872118

3. Singh, J. (11 de Dezembro de 2017). *Sentiment analysis using machine learning classifiers*. Obtido de SpringerOpen: https://hcis-journal.springeropen.com/articles/10.1186/s13673-017-0116-3