Instituto Superior de Engenharia do Porto

Licenciatura em Engenharia Informática

3DI

Relatório de Algoritmia Avançada

Grupo 051

Por Beatriz Seixas (1190424),

Jéssica Alves (1190682)

Pedro Santos (1190967),

Tiago Costa (1191460)

Porto, janeiro de 2022

Índice geral

1. INTRODUÇÃO

Na sequência do projeto referente à unidade curricular de *ALGAV* foi proposto o desenvolvimento de várias *user stories* com fim a criar uma aplicação cuja missão é fornecer aplicações de manipulação e visualização de grafos de redes sociais.

Neste documento iremos apresentar uma breve explicação do problema, assim como o pensamento para cada *user story*, e exemplos demonstrativos.

As competências requisitadas para este sprint, foram as seguintes:

- Consideração de estados emocionais para encontrar caminhos
- Sugestão de grupos
- Calculo de novos valores dos estados emocionais
- Estado de arte- Lingua Natural

2. CONSIDERAÇÃO DE ESTADOS EMOCIONAIS PARA ENCONTRAR CAMINHOS

2.1. Funcionalidade desenvolvida por Todos os Elementos do Grupo

Aos predicados *A-Star, Best First e DFS*, foi adicionado um parâmetro denominado de *EmotionBool*. Este parâmetro identifica se a execução do predicado irá ter em conta os estados emocionais de um utilizador ou não. Sendo os valores possíveis para este parâmetro, 0 para ignorar os estados emocionais e 1 para considerar esse aspecto na procura do caminho. Para isso, o predicado *emotion_checkSameEmotion*/3 é executado para o jogador atual e o seu amigo que está a ser considerado para o caminho, onde este irá verificar se é necessário fazer a consideração ou não. Caso a consideração dos estados emocionais seja necessária, chamando o *emotion_getMax*/3 que, para um determinado jogador, retorna a emoção com o valor mais elevado. Assim, é possível obter o estado emocional mais elevado para cada um dos jogadores a ser considerados, sendo as suas duas emoções são comparadas e, caso estas sejam, permite continuar com o algoritmo.

Um exemplo prático desta implementação é a implementação deste método no predicado A-Star do jogador com id de 1 para o com id de 5, onde inicialmente quando os estados emocionais não são considerados, este retorna o caminho suposto, sendo este [1,2,5]. Por outro lado, quando se define para considerar os estados emocionais, tendo o jogador atual um estado emocional diferente (neste caso, angústia) do seu amigo (alívio) com o id número 2, este já não encontra nenhum caminho devido ao seu único amigo não ter o mesmo estado emocional que o próprio utilizador. Para este exemplo foi utilizada a base de conhecimento parcial.

```
emotion_getMax(PlayerId, MaxValue, MaxEmotion):-
    occ(PlayerId, Joy, Anguish, Hope,Deception, Fear, Relief),
    emotion_maxEmotion([Joy, Anguish, Hope, Deception, Fear, Relief], [joy, anguish, hope, dec
eption, fear, relief],-100, emotion, MaxValue, MaxEmotion).

emotion_maxEmotion([], [], MaxAuxValue, MaxAuxEmotion, MaxReturnValue, MaxReturnEmotion):- !,
MaxReturnValue is MaxAuxValue, MaxReturnEmotion = MaxAuxEmotion.
emotion_maxEmotion([HV|TV], [HE|TE], MaxAuxValue, _, MaxReturnValue, MaxReturnEmotion):-
    HV >= MaxAuxValue, !,
    emotion_maxEmotion(TV, TE, HV, HE, MaxReturnValue, MaxReturnEmotion).
emotion_maxEmotion([_|TV], [_|TE], MaxAuxValue, MaxAuxEmotion, MaxReturnValue, MaxReturnEmotio
n):-
    !, emotion_maxEmotion(TV, TE, MaxAuxValue, MaxAuxEmotion, MaxReturnValue, MaxReturnEmotion
).

emotion_checkSameEmotion(EmotionBool, Act, X):-
    ((EmotionBool =:= 1, !,
     emotion_getMax(Act, _, ActEmotion),
     emotion_getMax(X, _, XEmotion),
     ((ActEmotion = XEmotion,!);
     false));
    true).
```

```
?- aStar_find(1,0,2,1,5,L,C).
L = [1, 2, 5],
C = 65.0 .

?- aStar_find(1,1,2,1,5,L,C).
false.
```

3. SUGESTÃO DE GRUPOS

3.1. Funcionalidade desenvolvida por Tiago Costa (1191460) e Beatriz Seixas (1190424)

Este predicado tem como objetivo de retornar o maior número de users com tags em comum.

A primeira variante, recebe como parâmetros: o id do utilizador, o número de tags que o utilizador quer, o número mínimo de utilizadores, a lista de tags, retornando a lista resultantes de tags e a lista de users com tags em comum.

```
common_tags(Id, NTags,NUsers,TagList,ResultTag,ResultUsers):-
    get_time(T1),
    node(Id,_,All_TagsT),
    common_tags_change_to_synonyms(All_TagsT, All_Tags),
    findall(Combination,common_tags_combination(NTags,All_Tags,Combination),CombinationsTemp),
    common_tags_test_list(CombinationsTemp, TagList, Combinations),
    findall(UserId,node(UserId,_,_),Users),
    asserta(common_tags_users([],[],0)),
    common_tags_users_combination(NTags,NUsers,Users,Combinations),
    common_tags_users(ResultTag,ResultUsers,_),
    retractall(common_tags_users(_,_,_)),
    write('Solution found in '),
    get_time(T2),
    T is T2-T1,write(T),write(' seconds'),nl.
```

Ao invés de pegarmos em todas as tags, vamos apenas pegar nas tags do user, trocar para sinónimos e encontramos todas as combinações.

```
common_tags(Id, NTags,NUsers,TagList,ResultTag,ResultUsers):-
    get_time(T1),
    node(Id,_,All_TagsT),
    common_tags_change_to_synonyms(All_TagsT, All_Tags),
    findall(Combination,common_tags_combination(NTags,All_Tags,Combination),CombinationsTemp),
    common_tags_test_list(CombinationsTemp, TagList, Combinations),
    findall(UserId,node(UserId,_,_),Users),
    asserta(common_tags_users([],[],0)),
    common_tags_users_combination(NTags,NUsers,Users,Combinations),
    common_tags_users(ResultTag,ResultUsers,_),
    retractall(common_tags_users(_,_,_)),
    write('Solution found in '),
    get_time(T2),
    T is T2-T1,write(T),write(' seconds'),nl.
```

No método abaixo verificamos se cada combinação de tags, contem as tags que o utilizador decidiu (caso tenha especificado), retornando apenas os grupos que contêm as tags especificadas pelo user.

```
common_tags_test_list([],_,[]):-!.
common_tags_test_list([CombinationsH|CombinationsT], Tags, FinalCombinations):-
    common_tags_test_list(CombinationsT, Tags, FinalCombinations1),
    (common_tags_test_lists(Tags, CombinationsH) ->
        append([CombinationsH], FinalCombinations1, FinalCombinations);
        append([], FinalCombinations1, FinalCombinations)).

common_tags_test_lists(List1, List2) :-
    forall(member(Element,List1), member(Element,List2)).
```

Por fim temos as condições que vão verificar se o tamanho de users no retorno é maior do que o indicado pelo utilizador, e caso se verifique, fazemos a validação para ver se o tamanho atual é superior ao tamanho guardado, caso seja, retiramos o valor guardado e guardamos o novo valor.

```
common_tags_users_combination(_,_,_,[]).
common_tags_users_combination(NTags,NUsers,Users,[Combination|Combinations]):-
    common_tags_users_combination_aux(NTags,Combination,Users,Users_With_Tags),
    common_tags_users_combination(NTags,NUsers,Users,Combinations),
    !,
    length(Users_With_Tags, L),
    ( L >= NUsers->
    common_tags_users(_,_,Size),
    (L > Size ->
    retract(common_tags_users(_,_,_)),
    asserta(common_tags_users(Combination,Users_With_Tags, L)) ; ! );!).
```

Um exemplo prático, com os parâmetros: 1, que é o id do jogador,3 que é o número de tags, 2 que representa o número mínimo de utilizadores, “natureza, música, sw” que são as tags escolhidas, “P” como a lista de tags e por fim “N” como a lista de users.

```
?- common_tags(1,3,2,[natureza,musica,sw],P, N).
Solution found in 0.0 seconds
P = [natureza, musica, sw],
N = [1, 13, 23, 200, 51].
```

4. CÁLCULO DE NOVOS VALORES DOS ESTADOS EMOCIONAIS

4.1. Funcionalidade Desenvolvida por Jéssica Alves (1190682) e Pedro Santos (1190967)

O predicado para o cálculo de novos valores dos estados emocionais de cada jogador apresenta duas variantes: uma para o cálculo das mesmas na alteração da força de relação (*emotion_relationChange*/4) e outra para o cálculo de novos valores para as emoções relativo a sugestão de grupos (*emotion_groupSuggestion*/4).

A primeira variante, recebe como parâmetros: o id do jogar atual, sendo este utilizado para ir pesquisar os valores dos estados emocionais atuais; o valor da força de relação, referente à diferença entre o número de *likes* e dislikes; e dois valores referentes ao retorno das emoções Alegria (*Joy*) e Angústria (*Anguish*) que serão as emoções afetadas por esta funcionalidade. Este obtém, através do id, os valores para a Alegria e Angústia do jogador atual, verifica se este é superior a zero (se este apresenta mais *likes* do que *dislikes*), se sim irá se aumentar a alegria e diminuir a angústia, devido a alteração positiva na força de relação. Caso o contrário aconteça, a angústia será aumentada e a alegria será diminuída.

```
emotion_relationChange(PlayerId, Value, NewJoy, NewAnguish):-
    occ(PlayerId, Joy, Anguish, _, _, _, _),
    ((Value > 0, !,
     emotion_increase(Joy, Value, 200, NewJoy),
     emotion_decrease(Anguish, Value, 200, NewAnguish));
    (Value < 0, !,
    emotion_increase(Anguish, -Value, 200, NewAnguish),
    emotion_decrease(Joy, -Value, 200, NewJoy))),
    retract(occ(PlayerId, Joy, Anguish, Hope, Deception, Fear, Relief)),
    asserta(occ(PlayerId, NewJoy, NewAnguish, Hope, Deception, Fear, Relief)).
```

Para os aumentos e diminuições de cada estado emocional, foram utilizadas duas fórmulas. A fórmula de aumento de uma emoção (*emotion_increase*/4), representada na equação seguinte, utiliza o valor anterior de uma determinada emoção, valores de saturação (no nosso caso 200 para a força de relação) e o valor da diferença de *likes* e *dislikes* passado por parâmetro, sendo a sua implementação visível na figura em baixo representada.

$$Emoção_{t+1} = Emoção_t + (1 - Emoção_t) * \frac{mínimo(Valor, Valor\_Saturação)}{Valor\_Saturação}$$

```
emotion_increase(PreviousValue, Value, Saturation, Return):-
    ((Value < Saturation, !, Min is Value);
    (!, Min is Saturation)),
    Div is Min / Saturation,
    Return is PreviousValue + (1 - PreviousValue) * Div.
```

Já na fórmula para a diminuição de uma emoção, os mesmos parâmetros serão utilizados, mas de uma forma diferente para obter a diminuição da mesma, sendo a sua implementação representada na figura seguinte (*emotion_decrease*/4). Após o cálculo das mesmas, o predicado irá retirar o facto *occ/7* do jogador atual e atualizá-lo com os novos valores.

$$Emoção_{t+1} = Emoção_t * (1 - \frac{mínimo(Valor, Valor\_Saturação)}{Valor\_Saturação})$$

```
emotion_decrease(PreviousValue, Value, Saturation, Return):-
    ((Value < Saturation, !, Min is Value);
    (!, Min is Saturation)),
    Div is 1 - (Min / Saturation),
    Return is PreviousValue * Div.
```

De outra forma, a segunda variante, responsável pelo cálculo das mesmas na sugestão de grupos, recebe por parâmetro os seguintes valores: o id do jogador atual (à semelhança do primeiro predicado); o número de *tags* pretendido, utilizado no método da sugestão de grupos; da mesma forma, o número de jogadores, também para passar por parâmetro para a sugestão de grupos; as tags obrigatórias com o mesmo propósito; e, por último, os valores de retorno para as emoções a recalcular neste predicado, sendo estas a

Esperança (Hope), Deceção (Deception), Medo (Fear) e Alívio (Relief). Para este predicado foram adicionados dois novos factos, *hope*/2 e fear/2, onde está representado o id do jogador e o jogador que este tem esperança ou medo que seja sugerido, assim cada jogador pode definir que tem medo de que outro seja sugerido, contribuindo para o par de emoções medo/alívio ou mesmo que espera que um jogador seja sugerido, afetando o par de emoções esperança/deceção. Assim, o método em questão chama o predicado *common_tags*/6, para obter o grupo de utilizadores sugeridos, sendo depois chamados os predicados *emotion_checkHope*/4 e *emotion_checkFear*/4 para ser feito o cálculo dos dois pares de emoções afetados.

```
emotion_groupSuggestion(PlayerId, TagCount, PlayerCount, MandatoryTags, NewHope, NewDeception,
 NewFear, NewRelief):-
    common_tags(PlayerId, TagCount, PlayerCount, MandatoryTags, _,SuggestedGroup),
    emotion_checkHope(PlayerId, SuggestedGroup, NewHope, NewDeception),
    emotion_checkFear(PlayerId, SuggestedGroup, NewFear, NewRelief),
    retract(occ(PlayerId, Joy, Anguish, _, _, _, _)),
    asserta(occ(PlayerId, Joy, Anguish, NewHope, NewDeception, NewFear, NewRelief)).
```

O predicado *emotion_checkHope*/4 recebe o id do jogador, a sugestão de grupo e os valores antigos para a esperança e deceção para percorrer a lista de jogadores sugeridos, através do predicado *emotion_countHope*/4, e contar quantos utilizadores que o jogador atual tinha esperança de que fossem sugeridos, identificados no facto *hope*/2. Este valor irá ser utilizado no quociente do novo cálculo da emoção, sendo este feito com número de jogadores desejados a dividir pelo número de jogadores sugeridos relativo à emoção esperança. Este valor é depois transformado, através do seu complementar, para a emoção deceção, por exemplo, se três jogadores dos que o utilizador esperasse que fossem sugeridos, estivessem presentes na lista de sugestão (tendo esta cinco elementos), o quociente do cálculo da emoção esperança seria 3/5, enquanto o da deceção seria 2/5 (3 – 5 = 2). Neste caso, como o numerador do quociente da esperança é maior do que o denominador da deceção, aumentamos a esperança e diminuímos a deceção. Se o contrário acontecesse, a deceção seria aumentada e a esperança diminuída. O mesmo se aplica à execução do predicado *emotion_checkFear*/4 que segue os mesmos princípios para o par de emoções medo/alívio. Por último, este retira o facto *occ*/7 do jogador atual e atualiza os seus valores com os novos calculados neste predicado.

```
emotion_checkHope(PlayerId, SuggestedGroup, NewHope, NewDeception):-
    emotion_countHope(PlayerId, SuggestedGroup, 0, Counter),
    occ(PlayerId, _, _, Hope, Deception, _, _),
    length(SuggestedGroup, Length),
    Complementary is Length - Counter,
    ((Counter > Complementary, !,
      emotion_increase(Hope, Counter, Length, NewHope),
     emotion_decrease(Deception, Complementary, Length, NewDeception));
    (emotion_increase(Deception, Complementary, Length, NewDeception),
    emotion_decrease(Hope, Counter, Length, NewHope))).
```

Um exemplo da utilização destes métodos, seria os exemplos em baixo representados onde calculamos os novos valores para a alegria e angústia relativos a uma diferença de *likes* e *dislikes* de -100, aumentando a angústia para 0.75 e diminuindo a alegria para 0.25. Por outro lado, na sugestão de grupos, para o utilizador com o id de 1 na base de conhecimento parcial, 2 como número de *tags* e jogadores, *a* e *b* como *tags* obrigatórias, obtemos novos valores para os dois pares de emoções relativos à sugestão de grupos. A deceção aumenta devido à nenhum dos jogadores esperados serem sugeridos, levando ao quociente da esperança ser 1/3 (a lista de sugestão apresenta 3 elementos) e, por sua vez, o da deceção ser 2/3, sendo o novo valor da deceção 0.83, e a esperança é diminuída com o quociente 1/3, levando ao seu novo valor de 0.33. O mesmo se aplica ao par de emoções medo/alívio, onde o quociente do medo é 1/3, devido a apenas 1 jogador que o utilizador atual definiu que tinha medo de este ser sugerido estar presente no grupo de sugestão, levando à sua diminuição. Por outro lado, o quociente do alívio é 2/3 devido ao cálculo complementar com o medo.

```
?- emotion_relationChange(1, -100, Joy, Anguish).
Joy = 0.25,
Anguish = 0.75 .
```

```
?- emotion_groupSuggestion(1,2,2,[a,b],Hope,Deception,Fear,Relief).
Solution found in 8.702278137207031e-5 seconds
Hope = Fear, Fear = 0.33333333333333337,
Deception = Relief, Relief = 0.8333333333333333 .
```

5. ESTADO DA ARTE – LÍNGUA NATURAL

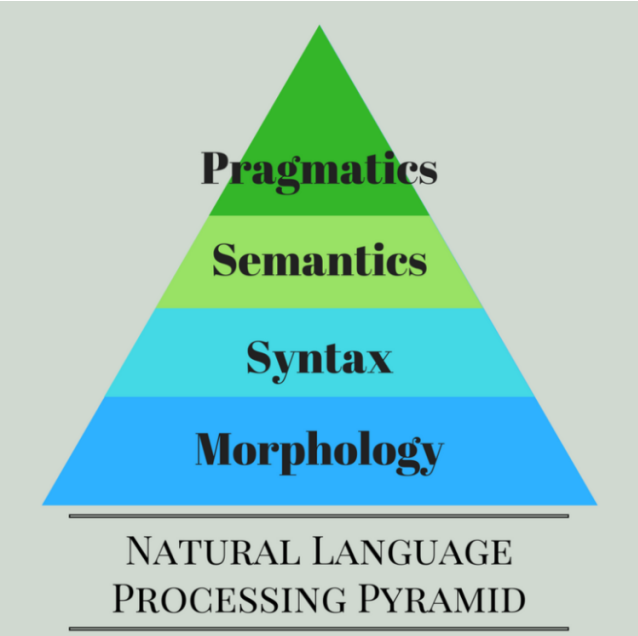
5.1. Funcionalidade desenvolvida por Todos os Elementos do Grupo

Este tópico tem como objetivo o estudo do tema "Processamento de Linguagem Natural: Análise Sentimental Aplicada a Redes Sociais", e foi realizado no âmbito da disciplina de ALGAV (Algoritmia Avançada).
Com este trabalho pretendemos mostrar o desenvolvimento da PLN (processamento de linguagem natural) nos aspetos emocionais das redes sociais.

O Processamento de Linguagem Natural (PLN) — Natural Language Processing (NLP) é uma vertente da Inteligência Artificial que tem como objetivo estudar a capacidade e limitações de uma máquina em entender a linguagem dos seres humanos.
Isto é, uma interface entre a linguagem homem-máquina. Dessa forma, o objetivo do PLN é fornecer aos computadores a capacidade de entender e compor textos.

Entender um texto muitas vezes pode ser complicado, por causa da ambiguidade da linguagem natural. Os seres humanos não têm tanta dificuldade em desvendar os duplos-sentidos já que têm algo chamado conhecimento cultural e experiência anterior.

Para as máquinas, é uma tarefa difícil reconhecer a presença de ambiguidades.
O desafio torna-se evidente quando nos lembramos que, muitas vezes, é necessário entender situações mais complexas como: reconhecer o contexto, tonalidade da voz, extrair informações, interpretar os sentidos, analisar sentimentos, realizar análise sintática, semântica, lexical e morfológica.
Podemos observar pela figura abaixo um exemplo resumido das tarefas de NLP sendo ilustrado em uma pirâmide em diferentes níveis

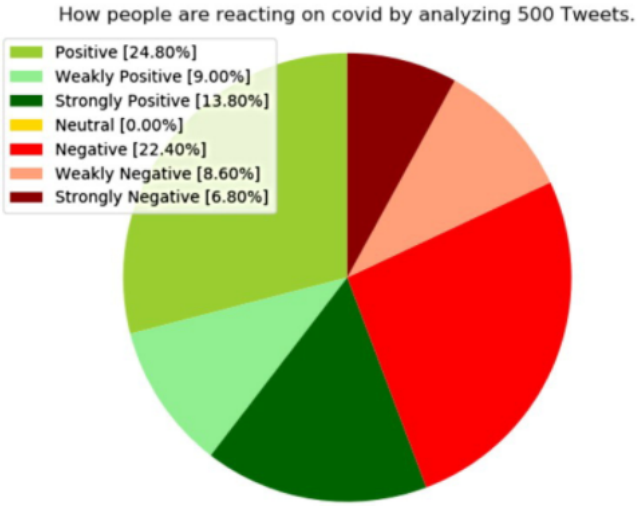


Nos últimos anos, as redes sociais online revolucionaram a comunicação interpessoal. Como sabemos as redes sociais são das principais fontes de dados contemporâneos. Nos dias de hoje as redes sociais têm desde posts, imagens, vídeos etc…
Com o acesso a linguagem natural, é possível que uma máquina processe a linguagem humana e também que seja compreendido o significado de uma mensagem.

Pesquisas recentes sobre análise de linguagem nas redes sociais têm se concentrado cada vez mais no impacto desta última em nossas vidas diárias, tanto em nível pessoal quanto profissional.
O processamento de linguagem natural (PLN) é um dos caminhos mais promissores para o processamento de dados de redes sociais.

É um desafio científico desenvolver métodos e algoritmos poderosos que extraiam informações relevantes de um grande volume de dados provenientes de múltiplas fontes e idiomas em vários formatos ou de forma livre.

Na seguinte imagem conseguimos observar os resultados de uma amostra de tamanho 500 tweets de um estudo realizado na rede social *Twitter* do impacto da palavra covid. Neste estudo foi usado uma RNN para classificar as emoções representadas nestes tweets.



As reações e avaliações de vários anúncios e decisões políticas, após o seu anúncio, provocam uma maior atividade por parte das pessoas que discutem e falam sobre os efeitos na rede social. Aumentando assim drasticamente o número de tweets relacionados com o tema.

Podemos concluir que os tweets em que a palavra covid está presente, existe um equilibrado misto de emoções positivas e negativas.

Mas existem diversos problemas neste tipo de tratamento de dados, como por exemplo, os problemas enfrentados a nível do léxico.

Estes estão relacionados com a identificação da semântica da palavra usada (Khan, 2016). As palavras e expressões com duplo significado dependem do contexto em que são utilizadas. Estas palavras não podem ser consideradas como positivas ou negativas sem haver um conhecimento do contexto). O léxico de opinião geral refere-se a palavras de opinião como bom, excelente, mau e pobre, etc.

Existe apenas um pequeno conjunto de léxicos de opinião à disposição do público. É necessário um léxico de opinião universal que forneça informação sobre todas essas palavras (Qiu et al. 2011). Uma técnica semi-automática de lidar com este pro-

blema é encontrar sinónimos e antónimos de seeds de léxicos inicialmente dadas, passados para o motor de busca. O processo é repetido várias vezes para explorar o maior número possível de palavras de opinião (Khan, 2016).

Concluindo assim o nosso trabalho podemos observar que o processamento de linguagens naturais é uma ferramenta bastante importante nos dias de hoje.

No entanto é evidente que ainda existem problemas com os métodos utilizados para o processamento de linguagem natural, como por exemplo, o problema das palavras com duplo sentido, realçando assim o espaço que existe para futura evolução no ramo da inteligência artificial.

6. CONCLUSÕES

Em suma, ao desenvolvermos os algoritmos requeridos nos casos de uso da disciplina de ALGAV, consolidámos aquilo que aprendemos ao longo do semestre, tendo-nos sido possível observar aplicações desta unidade curricular num projeto de média/grande escala que poderia vir a ser utilizado no mundo real.