Relevância de Textos com Lógica Fuzzy

Eric Calasans de Barros *José Genilson da Silva Filho*

Universidade Federal do Rio Grande do Norte

4 de dezembro de 2017

Introdução

- **Natural Language Processing(NLP)**
- **Classificação de Texto**
- **Perspectivas de Personalidade**
- **Classificação de Emoção**

Metodologia

Ferramentas Utilizadas

- **Python:**
  `nltk` - para análise e processamento do texto
  `skfuzzy` - para calcular relevância do texto através da lógica fuzzy
- **Textos para validação** - `futebol.txt` e `medicina.txt`

Metodologia

```
1 stemmer = RSLPStemmer() #extracao dos radicais das
      palavras
2 palavras = ['jogador', 'futebol'] # dicionario
3 arquivo = open('medicina.txt', 'r') #abrindo o arquivo
4 texto = arquivo.read() \# texto a ser comparado
5 words = (nltk.word_tokenize(texto)) #texto tokenrizsdo
6 stops = set(stopwords.words("portuguese")) #conjunto de
      palavras sem importancia para a classificacao do
      texto
7 word_features = ([w for w in words if not w in stops])
     #stop words removioda do texto
```

Listing 1: Preparação do Texto

Metodologia

Cálculo dos Matches

- **Match Total**

$$matchTotal_i = \frac{nOcorrCompleta_i}{\sum_{i=1}^{n} nOcorrCompleta_i}$$

- **Match Radical**

$$matchRadical_i = \frac{nOcorrRad_i}{\sum_{i=1}^{n} nOcorrRadi}$$

- **No Match**

$$noMatch_i = \frac{ausRad_i}{\sum_{i=1}^{n} ausRad_i}$$

Metodologia

```
1  #Procurando match total
2  def matchWord(texto, dicionario):
3    bag_words ={} # Inicia o dicionario que vai guardar
       os matchs do dicionario no texto
4    for i in dicionario:
5      bag_words[i] = 0
6  # Procura os matchs entre o dicionario e o texto;
       incrementa +1 quando encontra na chave do
       dicionario que
7  # se refere a palavra que esta sendo analisada
8    for palavraTexto in texto:
9      for palavraDicionario in dicionario:
10       if (palavraTexto == palavraDicionario):
11         bag_words[palavraDicionario] +=1
12
13   return(bag_words)
```

Listing 2: Match total

Metodologia

```
1  # Procurando match de radical
2  def macthRadical(texto, dicionario):
3    textoStemmer = [stemmer.stem(w.lower()) for w in
       texto] # Transforma as palavras do dicionario em
       radicais
4    dicStermmer = [stemmer.stem(w.lower()) for w in
       dicionario] # Transforma as palavras do texto em
       radicais
5    bag_words ={}
6    nMatch = {}  # Palavras que nao concordam com o
       radical
7   # Inicia o dicionario que sera usado para
       contabilizar os radicais e as palavras que nao
       tiveram match
8    for i in dicStermmer:
9      bag_words[i] = 0
10     nMatch[i] = 0
```

Listing 3: Match de radical

Metodologia

```
# Procura os radicais como feito no metodo anterior
for palavraTexto in textoStemmer:
  for palavraDicionario in dicStermmer:
    if (palavraTexto == palavraDicionario):
      bag_words[palavraDicionario] += 1

# Procura as palavras que nao estao presente no texto
   e estao no dicionario
for palavraDicionario in dicStermmer:
  if (not palavraDicionario in textoStemmer):
    nMatch[palavraDicionario] += 1

return(bag_words, nMatch)
```

Listing 4: Match de radical(cont.)

Metodologia

```
#Relevancia de uma palavra
def relevancia():
  #Inicializacoes
   total = matchWord(word_features, palavras)
   radical, nMatch = macthRadical(word_features,
     palavras)

   relevancia_total_sum = 0
   relevancia_radical_sum = 0
   relevancia_nMatch_sum = 0

```

Listing 5: Relevância de cada palavra

Metodologia

```
1  for matchPalavra, key in enumerate(total):
2    if(relevancia_total_sum == 0):
3      relevancia_total_sum = 1
4    relevancia_total[key] = total[key]/
     relevancia_total_sum
5
6  for matchPalavra, key in enumerate(radical):
7    if(relevancia_radical_sum == 0):
8      relevancia_radical_sum = 1
9    relevancia_radical[key] = radical[key]/
     relevancia_radical_sum
```

Listing 6: Relevância de cada palavra(cont.)

Metodologia

```
1  for matchPalavra, key in enumerate(nMatch):
2    if(relevancia_nMatch_sum == 0):
3      relevancia_nMatch[key] = 0
4    else:
5      if(relevancia_nMatch_sum == 0):
6        relevancia_nMatch_sum = 1
7      relevancia_nMatch[key] = nMatch[key]/
     relevancia_nMatch_sum
```

Listing 7: Relevância de cada palavra(cont.)

Metodologia

```
1   print(relevancia_total)
2   print(relevancia_radical)
3   print(relevancia_nMatch)
4
5   return(relevancia_total, relevancia_radical,
      relevancia_nMatch)
```

Listing 8: Relevância de uma palavra(cont.)

Metodologia

```
1 def fuzzyRelText(relTotal=0, relRadical=0, relNoMatch
    =0):
2   # Cria as variaveis fuzzy:  Antecedentes e
    Consequente
3   # Antecedentes
4   total = ctrl.Antecedent(np.arange(start=0, stop=1.1,
    step=0.1), 'Total')
5   radical = ctrl.Antecedent(np.arange(start=0, stop
    =1.1, step=0.1), 'Radical')
6   noMatch = ctrl.Antecedent(np.arange(start=0, stop
    =1.1, step=0.1), 'NoMatch')
7
8   #Consequente
9   nivelRelevancia = ctrl.Consequent(np.arange(start
    =0.0, stop=1.1, step=0.1), 'Relevancia')
```

Listing 9: Lógica Fuzzy

Metodologia

```
#Funcoes
#Total
total['nRelevante'] = fuzzy.trimf(total.universe, [0, 0, 0.8])
total['relevante'] = fuzzy.trimf(total.universe, [0.2, 1, 1])

#Radical
radical['nRelevante'] = fuzzy.trimf(radical.universe, [0, 0, 0.8])
radical['relevante'] = fuzzy.trimf(radical.universe, [0.2, 1, 1])
```

Listing 10: Lógica Fuzzy

Metodologia

```
1  #NoMatch
2  noMatch['nRelevante'] = fuzzy.trimf(noMatch.universe,
       [0, 0, 0.8])
3  noMatch['relevante'] = fuzzy.trimf(noMatch.universe,
     [0.2, 1, 1])
4
5  #resultado
6  nivelRelevancia['poucoRelevante'] = fuzzy.trimf(
     nivelRelevancia.universe, [0, 0, 0.4])
7  nivelRelevancia['relevante'] = fuzzy.trimf(
     nivelRelevancia.universe, [0.1, 0.5, 0.9])
8  nivelRelevancia['muitoRelevante'] = fuzzy.trimf(
     nivelRelevancia.universe, [0.6, 1, 1])
```

Listing 11: Lógica Fuzzy(cont.)

Metodologia

```
# Regras
r1 = ctrl.Rule(antecedent=total['relevante'] &
  radical['relevante'] & noMatch['nRelevante'],
        consequent=nivelRelevancia['muitoRelevante'])
r2 = ctrl.Rule(antecedent=total['nRelevante'] &
  radical['nRelevante'] & noMatch['relevante'],
        consequent=nivelRelevancia['poucoRelevante'])
r3 = ctrl.Rule(antecedent=total['relevante'] &
  radical['relevante'] & noMatch['nRelevante'],
        consequent=nivelRelevancia['relevante']%0.8)
r4 = ctrl.Rule(antecedent=total['relevante'] &
  radical['relevante'] & noMatch['relevante'],
        consequent=nivelRelevancia['relevante']%0.6)
```

Listing 12: Lógica Fuzzy(cont.)

Metodologia

```
# Regras
r5 = ctrl.Rule(antecedent=total['relevante'] &
  radical['nRelevante'] & noMatch['nRelevante'],
        consequent=nivelRelevancia['relevante']%0.7)
r6 = ctrl.Rule(antecedent=total['nRelevante'] &
  radical['relevante'] & noMatch['nRelevante'],
        consequent=nivelRelevancia['relevante']%0.5)
r7 = ctrl.Rule(antecedent=total['nRelevante'] &
  radical['nRelevante'] & noMatch['nRelevante'],
   consequent=nivelRelevancia['poucoRelevante'])
```

Listing 13: Lógica Fuzzy(cont.)

Metodologia

```
#Cria maquina de inferencia
controleRelevancia = ctrl.ControlSystem([r1, r2, r3,
  r4, r5, r6, r7])

#Prepara a simulacao
resultado = ctrl.ControlSystemSimulation(
  control_system=controleRelevancia)

#Entrada de dados
resultado.input['Total'] = relTotal
resultado.input['Radical'] = relRadical
resultado.input['NoMatch'] = relNoMatch
```

Listing 14: Lógica Fuzzy(cont.)

Metodologia

```
1  #Defuzzificacao
2  resultado.compute()
3
4  #Retorna relevancia
5  return resultado.output['Relevancia']
```

Listing 15: Resultado final

Metodologia

```
(total, radical, nMatch) = relevancia()
rele = 0

# A relevancia final do texto e calculada somando as
    relevancias de cada palavra do dicionario dividido
    pela quantidade
# de palavras no dicionario
for word, key in enumerate(total):
  rele += fuzzyText.fuzzyRelText(total[key], radical[
    stemmer.stem(key)], nMatch[stemmer.stem(key)])

print(rele/len(total))
```

Listing 16: Lógica Fuzzy(cont.)

Resultados

Texto `futebol.txt`

Há relatos de um esporte muito parecido com o futebol, embora usava-se muito a violência. O Soule ou Harpastum era praticado na Idade Média por militares que se dividiam em duas equipes: atacantes e defensores. Era permitido usar socos, pontapés, rasteiras e outros golpes violentos. Há relatos que mostram a morte de alguns jogadores durante a partida. Cada equipe era formada por 27 jogadores, onde grupos tinham funções diferentes no time: corredores, dianteiros, sacadores e guarda-redes. Na Itália Medieval apareceu um jogo denominado gioco del calcio. Era praticado em praças e os 27 jogadores de cada equipe deveriam levar a bola até os dois postes que ficavam nos dois cantos extremos da praça. A violência era muito comum, pois os participantes levavam para campo seus problemas causados, principalmente por questões sociais típicas da época medieval. O barulho, a desorganização e a violência eram tão grandes que o rei Eduardo II teve que decretar uma lei proibindo a prática do jogo, condenando a prisão os praticantes. Porém, o jogo não terminou, pois integrantes da nobreza criaram uma nova versão dele com regras que não permitiam a violência. Nesta nova versão, cerca de doze juízes deveriam fazer cumprir as regras do jogo.

Resultados

- **Dicionário:** `jogador; futebol`
- **Texto:** `futebol.py`
- **Matches:**

'futebol' $\rightarrow$ *total* $= 1.0$, *radical* $= 0.125$ e *noMatch* $= 0$

'jogador' $\rightarrow$ *total* $= 0.0$, *radical* $= 0.875$ e *noMatch* $= 0$

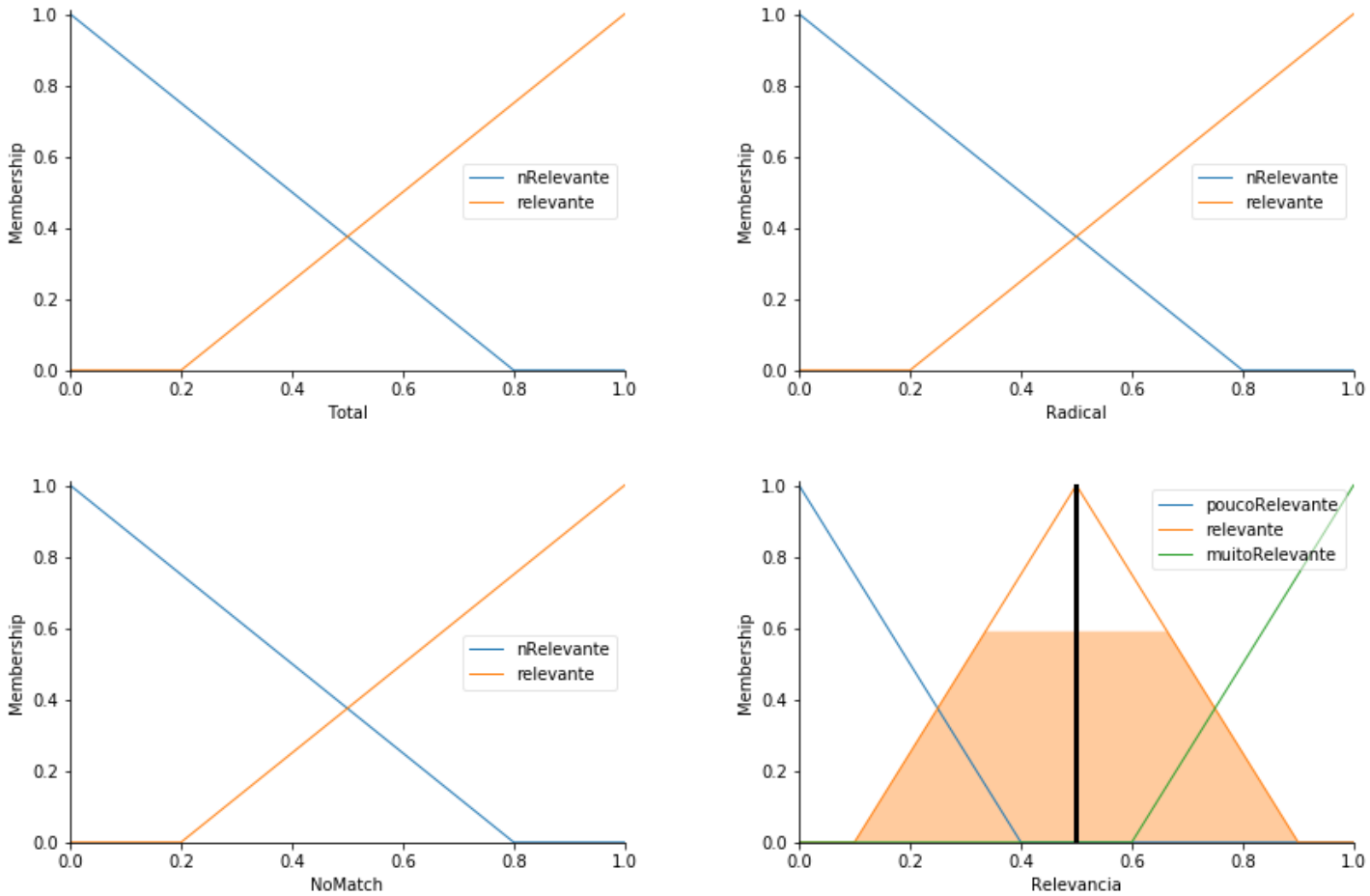

Figura: Palavra 'futebol' no texto `futebol.txt`

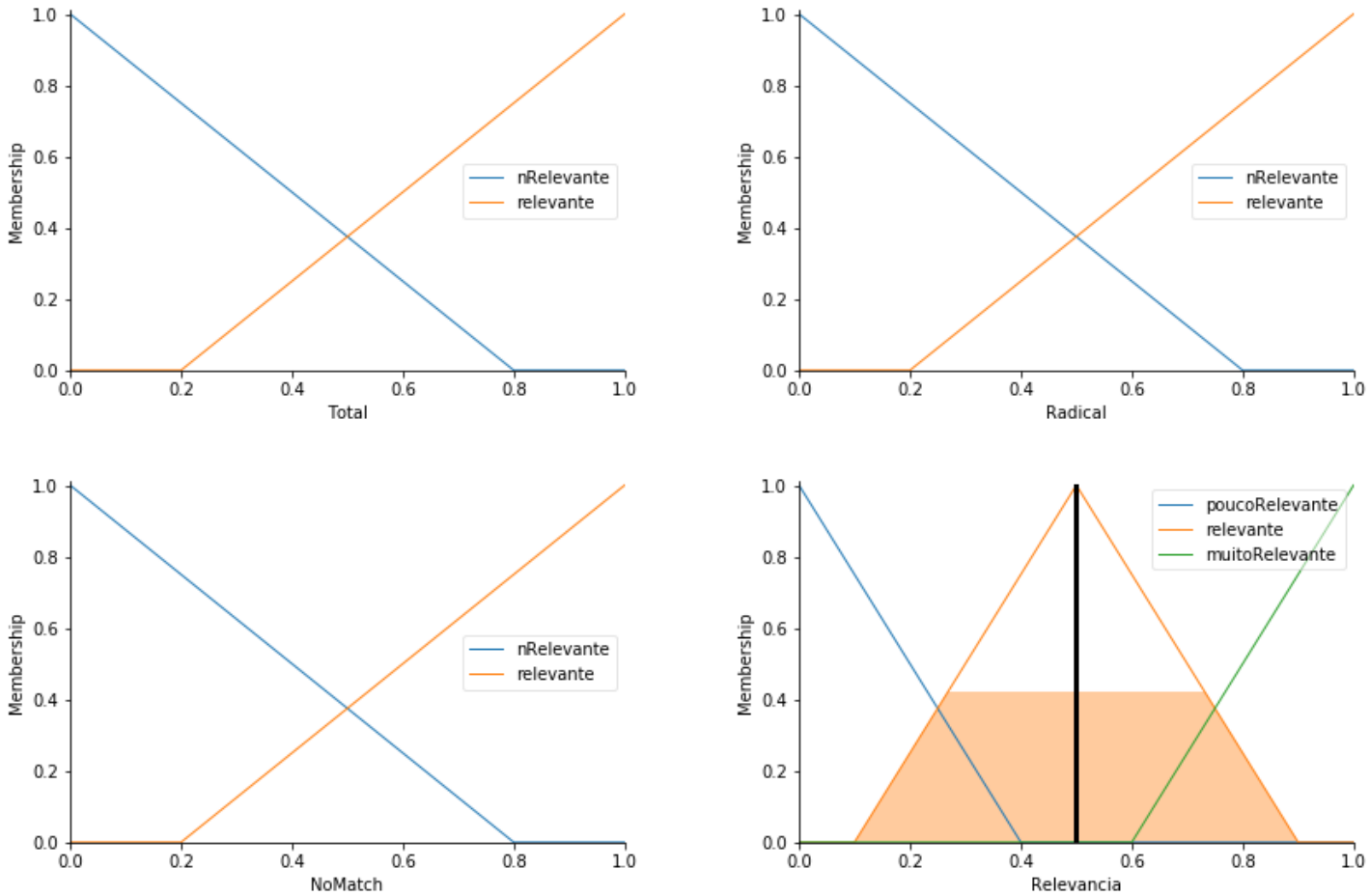

Figura: Palavra 'jogador' no texto `futebol.txt`

Resultados

Texto `medicina.txt`

A medicina é uma das muitas áreas do conhecimento ligada à manutenção e restauração da saúde. Ela trabalha, num sentido amplo, com a prevenção e cura das doenças humanas e animais num contexto médico. Ações de saúde pública e ambiental, incluindo a saúde animal, promoção, prevenção, controle, erradicação e tratamento das doenças, traumatismos ou qualquer outro agravo à integridade e bem-estar animais, além do controlo de sanidade dos produtos e subprodutos de origem animal para o consumo humano e animal compreendem a área da medicina da responsabilidade do profissional de saúde médico veterinário. Em Portugal, a saúde oral, higiene, integridade dentária, a sua limpeza e profilaxia compreendem a área da medicina da responsabilidade do Médico Dentista, que é um profissional da saúde capacitado na área de odontologia, e apesar de ter um âmbito de acção semelhante, não deve ser confundido com o Médico estomatologista. Porém no Brasil, odontologia e medicina são profissões distintas. Segundo a Organização Mundial da Saúde, saúde não é apenas a ausência de doença. Consiste no bem-estar físico, mental, psicológico e social do indivíduo.

Resultados

É um estado cumulativo, que deve ser promovido durante toda a vida, de maneira a assegurar-se de que seus benefícios sejam integralmente desfrutados em dias posteriores. Nesse contexto, diretrizes de organizações supra-nacionais compostas por eminentes intelectuais do globo relacionados à área de saúde estabeleceram um novo paradigma de abordagem em medicina. O santo patrono da Medicina é São Lucas.

Resultados

- **Dicionário:** `jogador; futebol`
- **Texto:** `medicina.py`
- **Matches:**

'futebol' $\rightarrow$ *total* $= 1.0$, *radical* $= 0.125$ e *noMatch* $= 0$

'jogador' $\rightarrow$ *total* $= 0.0$, *radical* $= 0.875$ e *noMatch* $= 0$

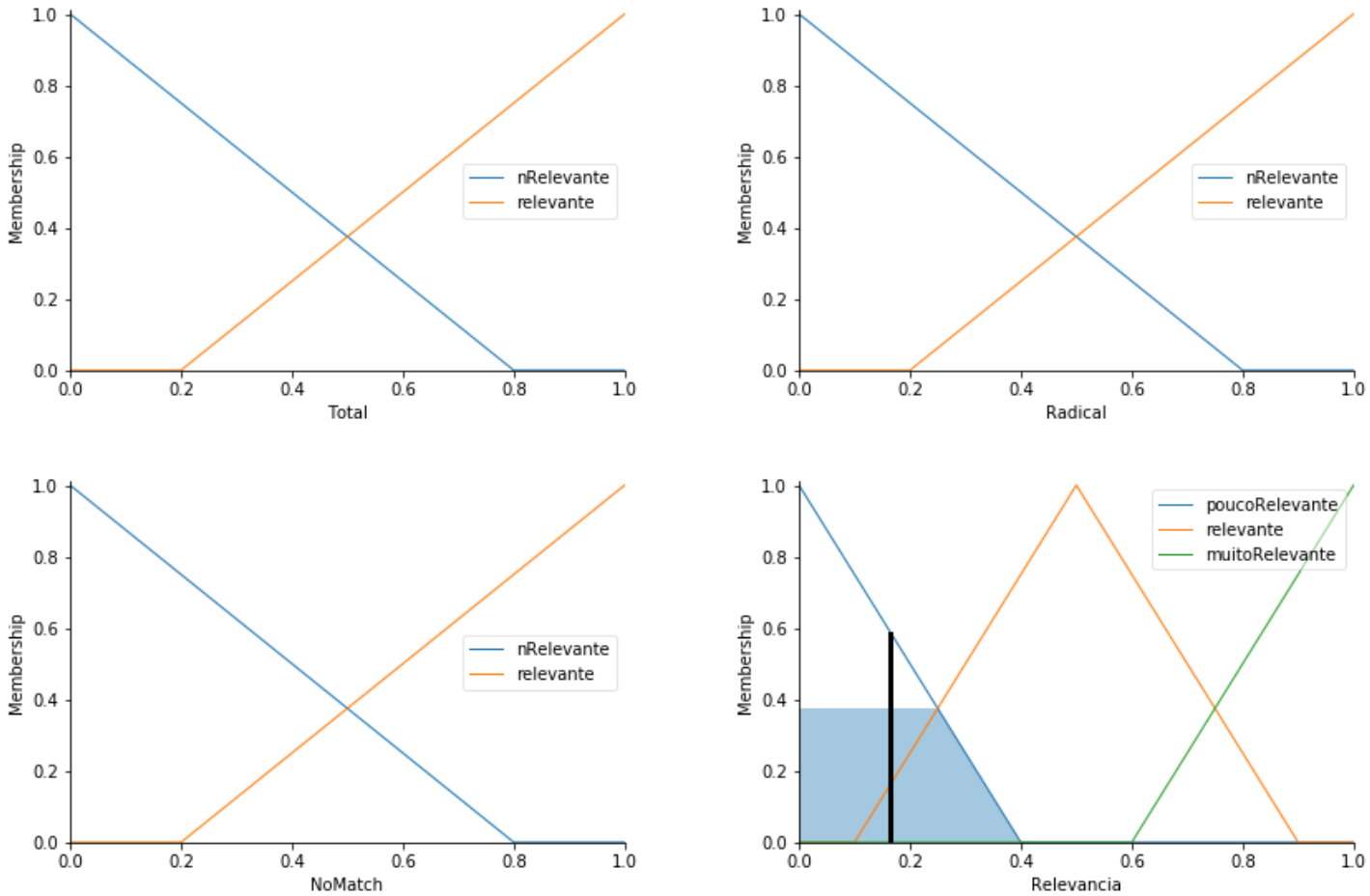

Figura: Palavra 'futebol' no texto `medicina.txt`

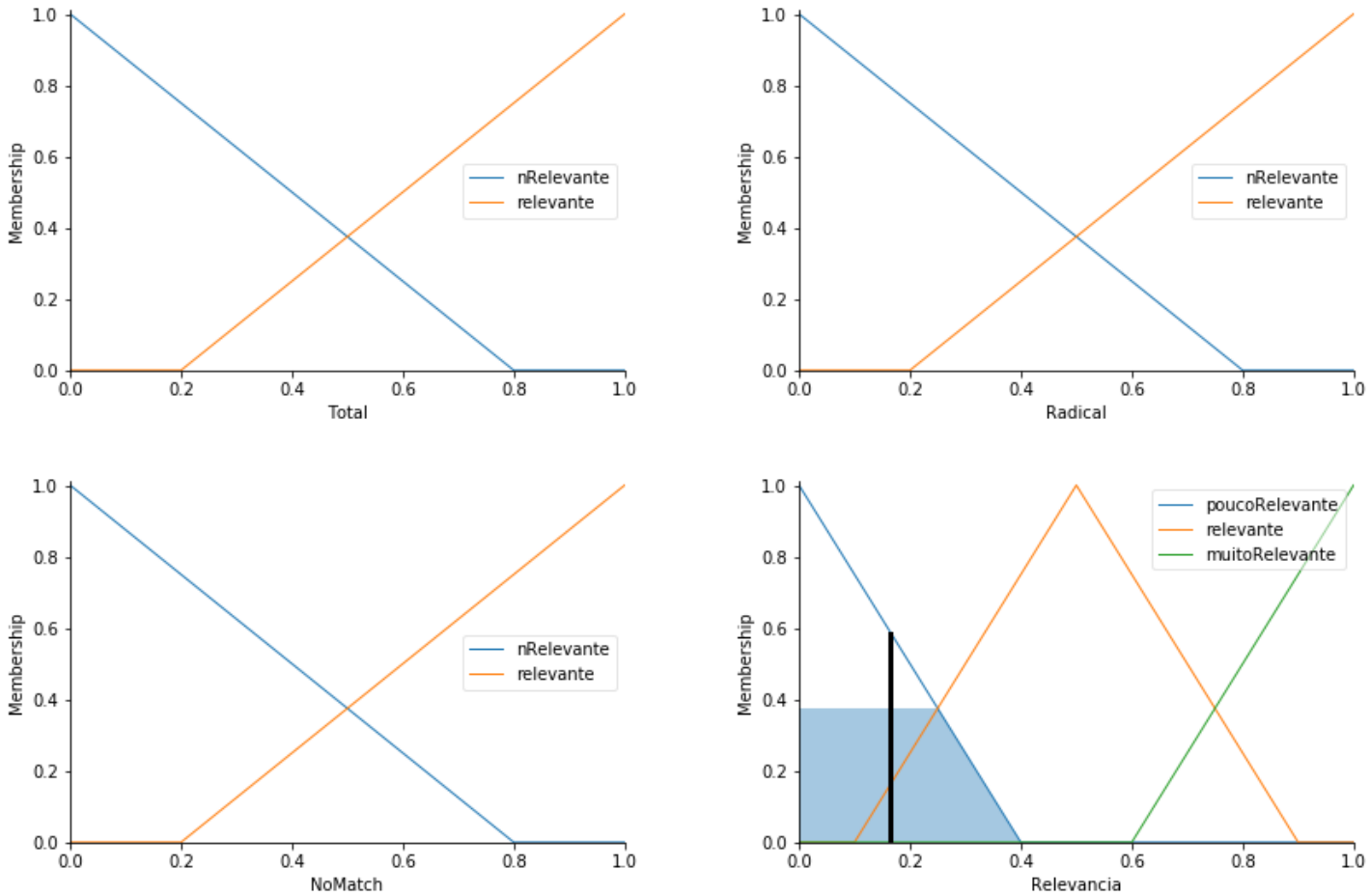

Figura: Palavra 'jogador' no texto `medicina.txt`

Conclusões