Projeto Wordnet: Documentação

Matheus Popst e Vitória Guardieiro

27 de junho de 2018

1 Arquivos

Para que as funções funcionem como esperado, é necessário que os arquivos *synsets.txt* e *hypernyms.txt* satisfaçam certas condições, especificadas tanto na página de Princeton quanto a seguir.

1.1 synsets.txt

Contém todos os synsets de substantivos da wordnet, um por linha, da seguinte maneira:

```
id,conjunto de sinônimos,definição
```

O `id` corresponde à linha do arquivo em que está descrito o synset, começando em zero, e o `conjunto de sinônimos` deve conter os sinônimos separados por espaços e palavras compostas como *American sign language* devem ser escritas com *underline* ao invés de espaços.

Um exemplo de linha deste arquivo é:

```
41,ASL American_sign_language,the sign language used in the United States
```

Não deve conter linhas em branco entre os dados.

1.2 hypernyms.txt

Contém todas as relações hiperonímia entre os *synsets*, da seguinte maneira:

```
id,hiperônimos
```

Assim como no arquivo *synsets.txt*, o campo `id` corresponde à linha do arquivo, começando em zero, e representa o *synset* que contém tal `id`. Os hiperônimos estão representados também por seu `id` e separados por vírgulas.

Um exemplo de linha deste arquivo é:

```
34,47569,48084
```

Também não deve conter linhas em branco entre os dados.

2 Funções principais

2.1 make-graph

Tendo os arquivos *synsets.txt* e *hypernyms.txt* na pasta, comece gerando a Wordnet, usando a função

```
(make-graph file-synsets file-hypernyms)
```

A função irá retornar uma lista de objetos synset, que contém: ID, nouns, hypernyms e definition. Em particular, o i-ésimo elemento da lista, é a entrada da wordnet de ID i. É importante ressaltar que é necessário salvar a wordnet com um `defvar`, por exemplo.

2.2 dictionary

A função `dictionary` tem a seguinte sintaxe:

```
(dictionary word wordnet)
```

Ela não retorna nada e imprime na tela os múltiplos significados daquela palavra. A palavra deve ser colocada entre áspas.

2.3 sca-words

A função `sca-words` (shortest-common-ancestor) tem a seguinte sintaxe:

```
(sca-words word1 word2 wordnet)
```

Novamente, as palavras devem ser strings, colocadas entre áspas. O retorno dela será uma tupla. No primeiro elemento conterá o **objeto** que é o menor ancestral comum entre as duas palavras e no segundo elemento, estará a soma da distância entre as palavras e o menor ancestral comum.

Um exemplo de chamada é:

```
CL-USER> (sca-words "summer" "apple" wordnet)
(#<SYNSET {1008FE70A3}> . 13)
```

2.4 distance

A função `distance` tem a seguinte sintaxe:

```
(distance word1 word2 wordnet)
```

É muito semelhante à anterior, com a diferença que retorna apenas a menor distância entre as duas palavras na wordnet, sem dizer qual é o menor ancestral comum.

Um exemplo de chamada é:

```
CL-USER> (distance "apple" "moon" wordnet)
8
```

2.5 outcast

A função `outcast` tem a seguinte sintaxe:

```
(outcast list-of-words wordnet)
```

Esta função irá retornar a palavra que tem a "menor semelhança", isto é, a palavra mais distante das outras.

Um exemplo de chamada é:

```
CL-USER> (outcast '("Russia" "Jesus" "Italy" "California") wordnet)
"Jesus"
```

3 Funções auxiliares

3.1 read-synsets

A função read-synsets tem a seguinte sintaxe:

```
(read-synsets file)
```

Esta função lê um arquivo que define os *synsets* da *wordnet* e retorna uma lista com os objetos *synsets* ordenados de forma crescente pelo `id`.

É utilizada para gerar a *wordnet* com a função `make-wordnet`.

3.2 find-synset

A função find-synset tem a seguinte sintaxe:

```
(find-synset word wordnet)
```

Esta função recebe uma palavra e a *wordnet* e retorna uma lista de *synsets* que contêm esta palavra.

Um exemplo de chamada é:

```
CL-USER> (find-synset "blue" wordnet)
(#<SYNSET {1007F61743}> #<SYNSET {1007F61123}> #<SYNSET {1007F60A63}>
 #<SYNSET {1007F60A43}> #<SYNSET {1007F60A23}> #<SYNSET {1007F60A03}>
 #<SYNSET {100782D843}>)
```

É utilizada na função `sca-words`.

3.3 go-up

A função go-up tem a seguinte sintaxe:

```
(go-up synset)
```

Esta função recebe um synset e retorna uma lista com tuplas contendo os synsets ancestrais dele, ou seja, os synsets que possuem path de hyperonômia partindo do synset, e a distância do ancestral a ele. Também recebe como paramêtro opcional `dist`, que é utilizado para a recursão.

Um exemplo de chamada é:

```
CL-USER> (go-up (car (find-synset "love" wordnet)))
((#<SYNSET {10098BCF03}> 0) (#<SYNSET {10098BD043}> 1)
 (#<SYNSET {1009A17843}> 2) (#<SYNSET {10099151E3}> 3)
 (#<SYNSET {10098EED03}> 4) (#<SYNSET {1009902123}> 5)
 (#<SYNSET {10099B29C3}> 6))
```

É utilizada na função `sca-synsets`.

3.4 sca-synsets

A função sca-synsets tem a seguinte sintaxe:

```
(sca-synsets synset1 synset2)
```

Esta função recebe dois objetos *synset* e retorna uma tupla contendo o objeto *synset* que é o ancestral comum mais próximo aos *synsets* dados e a soma da distância de cada *synset* a ele.

Um exemplo de chamada é:

```
CL-USER> (sca-synsets (nth 3100 wordnet) (nth 129 wordnet))
(#<SYNSET {1009902123}> . 9)
```

É utilizada na função `sca-synsets`.