Trabalho 3 – Similaridade Semântica

Descrição

Um tipo de questão encontrada no Teste de Inglês como Língua Estrangeira (TOEFL) é a "Questão de Sinônimo", onde os alunos são solicitados a escolher o sinônimo de uma palavra em uma lista de alternativas. Por exemplo:

```
 1. vexed                                          (Answer: (a) annoyed)
(a) annoyed
(b) amused
(c) frightened
(d) excited
```

Para esta tarefa, você construirá um sistema inteligente que pode aprender a responder a perguntas como esta. Para fazer isso, o sistema aproximará a semelhança semântica de qualquer par de palavras. A semelhança semântica entre duas palavras é a medida da proximidade de seus significados. Por exemplo, a semelhança semântica entre "carro" e "veículo" é alta, enquanto que entre "carro" e "flor" é baixa.

Para responder à pergunta do TOEFL, você calculará a semelhança semântica entre a palavra que recebeu e todas as respostas possíveis e escolherá a resposta com a maior semelhança semântica para a palavra dada. Mais precisamente, dada uma palavra *w* e uma lista de sinônimos potenciais s_1, s_2, s_3, s_4, calculamos as semelhanças de $(w; s_1)$, $(w; s_2)$, $(w; s_3)$, $(w; s_4)$ e escolhemos a palavra cuja semelhança com *w* é a maior.

Mediremos a similaridade semântica de pares de palavras calculando primeiro um *vetor descritor semântico* de cada uma das palavras e, em seguida, tomando a medida de similaridade como sendo a *similaridade de cosseno* entre os dois vetores.

Dado um texto com *n* palavras denotadas por $(w_1; w_2; \ldots ; w_n)$ e uma palavra *w*, $desc_w$ é o vetor descritor semântico de *w* calculado usando o texto. $desc_w$ é um vetor *n-dimensional*. A *i-ésima* coordenada de $desc_w$ é o número de sentenças nas quais *w* e w_i ocorrem. Por uma questão de eficiência, **armazenaremos o vetor descritor semântico como um dicionário**, não armazenando os zeros que correspondem a palavras que não coocorrem com *w*. Por exemplo, suponha que recebamos o seguinte texto (a abertura de *Notes from the Underground*, notas do subsolo, de Fyodor Dostoyevsky, traduzido por Constance Garnett):

I am a sick man. I am a spiteful man. I am an unattractive man. I believe my liver is diseased. However, I know nothing at all about my disease, and do not know for certain what ails me.

A palavra "man" só aparece nas três primeiras frases. Seu vetor descritor semântico seria:

```
{"i": 3, "am": 3, "a": 2, "sick": 1, "spiteful": 1, "an": 1, "unattractive": 1}
```

A palavra "liver" ocorre apenas na segunda frase, então seu vetor descritor semântico é:

```
{"i": 1, "believe": 1, "my": 1, "is": 1, "diseased": 1}
```

Armazenamos todas as palavras em letras minúsculas, uma vez que não consideramos, por exemplo, "Man" e "man" como palavras diferentes. No entanto, consideramos, por exemplo,

"believe" e "believes" ou "am" e "is" como palavras diferentes. Descartamos todas as pontuações.

A similaridade do cosseno entre dois vetores $u = \{u_1; u_2; ...; u_N\}$ e $v = \{v_1; v_2; ...; v_N\}$ é definido como:

$$\text{sim}(u, v) = \frac{u \cdot v}{|u||v|} = \frac{\sum_{i=1}^{N} u_i v_i}{\sqrt{\left(\sum_{i=1}^{N} u_i^2\right)\left(\sum_{i=1}^{N} v_i^2\right)}}$$

Não podemos aplicar a fórmula diretamente aos nossos descritores semânticos, uma vez que não armazenamos as entradas que são iguais a zero. No entanto, ainda podemos calcular a semelhança do cosseno entre os vetores considerando apenas as entradas positivas.

Por exemplo, a semelhança de cosseno de "man" e "liver", dados os descritores semânticos acima, é:

$$\frac{3 \cdot 1 \ (para\ a\ palavra\ "i")}{\sqrt{(3^2 + 3^2 + 2^2 + 1^2 + 1^2 + 1^2 + 1^2)(1^2 + 1^2 + 1^2 + 1^2 + 1^2)}} = {}^{3}/_{\sqrt{130}} = \ 0.2631 \ldots$$

Implementação

Implemente as seguintes funções em sinonimos.py. Observe que os nomes das funções diferenciam maiúsculas de minúsculas e não devem ser alterados. Você não tem permissão para alterar o número de parâmetros de entrada, nem para adicionar quaisquer variáveis globais. Isso fará com que seu código falhe ao ser executado nos testes, de forma que você não obterá nenhuma pontuação da funcionalidade. Você pode criar funções auxiliares. É fornecida a você uma versão inicial de sinonimos.py.

A. similaridade_cosseno(dict_1, dict_2):
   Esta função retorna a similaridade de cosseno entre os vetores esparsos dict_1 e dict_2, armazenados como dicionários.
   Por exemplo, similaridade_cosseno({"a": 1, "b": 2, "c": 3}, {"b": 4, "c": 5, "d": 6}) deve retornar aproximadamente 0.7.
B. constroi_descritores_semanticos(frases):
   Esta função recebe uma lista de frases que contém listas de palavras e retorna um dicionário d tal que para cada palavra w que aparece em pelo menos uma das frases, d[w] é em si um dicionário que representa o descritor semântico de w (obs: o nome da variável aqui é aleatório). Por exemplo, se frases representa a abertura de *Notes from the Underground* acima:

```
[["i", "am", "a", "sick", "man"],
 ["i", "am", "a", "spiteful", "man"],
 ["i", "am", "an", "unattractive", "man"],
 ["i", "believe", "my", "liver", "is", "diseased"],
 ["however", "i", "know", "nothing", "at", "all", "about", "my",
 "disease", "and", "do", "not", "know", "for", "certain", "what", "ails", "me"]]
```

   Parte do dicionário retornado deve ser:

```
{ "man": {"i": 3, "am": 3, "a": 2, "sick": 1, "spiteful": 1, "an": 1,
          "unattractive": 1},
  "liver": {"i": 1, "believe": 1, "my": 1, "is": 1, "diseased": 1},
  ... }
```

Com tantas chaves quantas palavras distintas apareçam na passagem.

C. constroi_descritores_semanticos_de_arquivos(nomes_arquivos):
Esta função pega uma lista de nomes de arquivos (nomes_arquivos) de strings, que contém os nomes dos arquivos, e retorna um dicionário dos descritores semânticos de todas as palavras nos contidas nos arquivos de nomes_arquivos, eles devem ser tratados como um único texto.
Você deve presumir que a seguinte pontuação sempre separa as frases: ".", "!", "?".
Você também deve presumir que essa é a única pontuação que separa as frases.
Suponha que apenas a seguinte pontuação esteja presente nos textos: [",", "-", "--", ":", ";"].

D. palavra_mais_similar(palavra, escolhas, descritores_semanticos, funcao_similaridade):
Esta função recebe uma palavra, uma lista de escolhas e um dicionário descritores_semanticos que é construído com constroi_descritores_semanticos e retorna o elemento de escolhas que tem a maior similaridade semântica com a palavra, com a similaridade semântica computada usando o dados em descritores_semanticos e a função de similaridade sendo funcao_similaridade. A função de similaridade é uma função que recebe dois vetores esparsos armazenados como dicionários e retorna um float. Um exemplo de tal função é similaridade_cosseno. Se a semelhança semântica entre duas palavras não puder ser calculada, ela será considerada -1. Em caso de empate entre vários elementos nas escolhas, aquele com o menor índice nas escolhas deve ser retornado (por exemplo, se houver um empate entre as escolhas[5] e escolhas [7], a escolhas[5] é retornada).

E. executa_teste_similaridade(nome_arquivo, descritores_semanticos, funcao_similaridade)
Esta função recebe um nome de arquivo (nome_arquivo) e retorna a porcentagem (ou seja, float entre 0,0 e 100,0) de questões nas quais palavra_mais_similar() adivinha a resposta corretamente usando a semântica descritores armazenados em descritores_semanticos, usando a função de similaridade funcao_similaridade.
O formato de arquivo é o seguinte. Em cada linha, recebemos uma palavra (em letras minúsculas), a resposta correta e as opções. Por exemplo, a linha:

feline cat dog cat horse

Representa a questão:

```
feline:
(a) dog
(b) cat
(c) horse
```

E indica que a questão correta é cat.
Por exemplo, executa_teste_similaridade pode ser utilizado como:

```
descritores_semanticos = constroi_descritores_semanticos_de_arquivos(["texto1.txt",
"texto2.txt"])
res = executa_teste_similaridade("teste.txt", descritores_semanticos,
similaridade_cosseno)
print(res, "das suposições estavam corretas")
```

Teste

Você deve testar sua implementação usando pequenos exemplos onde você conhece o resultado correto.

Você também deve tentar executar sua implementação com os romances recomendados (juntos): Guerra e Paz e O caminho de Swann, disponíveis no Projeto Gutenberg:

http://www.gutenberg.org/files/2600/2600-0.txt

http://www.gutenberg.org/cache/epub/7178/pg7178.txt

Coloquei os textos no arquivo do trabalho também. Você deve esperar obter suposições corretas entre 67,5% e 72,5% quando todas as funções forem implementadas corretamente.

Veja o arquivo saida_print.txt, que tem o que foi impresso no terminal no __main__ .