Семантическая близость слов на основе коллекции тестов

Лебедев Андрей, группа 210

Задача

Использовать предобученную модель на основе fasttext для подсчета близостей слов в датасетах wordsim similarity и wordsim relatedness, посчитать корреляцию с человеческими оценками из датасетов с помощью корреляции Спирмена.

Подключения

1) Модуль io для импортирования векторов слов из файла crawl-300d-2M.vec
2) Модуль collections для использования функции defaultdict()
3) Модуль scipy.stats для использования функции spearmanr() подсчета корреляции Спирмена

План работы

В словаре word_vec будем хранить множество слов из датасетов, которым в соответствие поставим векторы из модели. d_rel и d_sim – словари, организованные следующим образом:
dictionary = { ‘pairs’: [word1, word2], [word1, word2], ...– пары слов из датасетов
‘val’: val1, val2, ... – человеческие оценки близости
‘cos’: cos1, cos2, ... – косинусы между векторами слов
}

```
word_vec = defaultdict(list)
d_rel = defaultdict(list)
readfile('wordsim_relatedness_goldstandard.txt', d_rel, word_vec)
d_sim = defaultdict(list)
readfile('wordsim_similarity_goldstandard.txt', d_sim, word_vec)
```

Функция readfile читает каждый из файлов wordsim_relatedness и wordsim_similarity, добавляет в соответствующий словарь dictionary['pairs'] пары слов, в dictionary['val'] оценки людей, а также добавляет слова в word_vec – множество всех слов.

```
def readfile(name, main_dict, dict_word_vec):
    f = open(name, 'r', encoding='utf-8', newline='\n', errors='ignore')
    main_dict['pairs'] = []
    main_dict['val'] = []
    main_dict['cos'] = []
    for s in f:
        word1, word2, val = s.split()
        main_dict['pairs'].append([word1, word2])
        main_dict['val'].append(val)
        dict_word_vec[word1] = []
        dict_word_vec[word2] = []
    f.close()
```

Далее вызываем функцию load_vectors:

```
load_vectors('crawl-300d-2M.vec', word_vec)
```

Описание функции:

```
def load_vectors(fname, dict_word_vec):
    fin = io.open(fname, 'r', encoding='utf-8', newline='\n',
errors='ignore')
    n, d = map(int, fin.readline().split())
    for line in fin:
        tokens = line.rstrip().split(' ')
        if dict_word_vec.get(tokens[0], 0) != 0:
            dict_word_vec[tokens[0]] = list(map(float, tokens[1:]))
```

Основой для функции служит код с https://fasttext.cc/docs/en/english-vectors.html, который переделан под наши задачи: мы копируем не все векторы из файла, а

только для тех слов, которые встречаются в датасетах, то есть в множестве word_vec, для этого мы его и создавали.

Далее для каждой пары вектором вычисляем косинус угла между ними, вызывая функцию cosines_calculating:

```
cosines_calculating(d_rel, word_vec)
cosines_calculating(d_sim, word_vec)
```

Описание функции cosines_calculating и вспомогательной cosine:

```
def cosine(u, v):
    numerator = 0
    sum1 = 0
    sum2 = 0
    for i in range(len(u)):
        numerator += u[i] * v[i]
        sum1 += u[i] ** 2
        sum2 += v[i] ** 2
    return numerator / ((sum1 ** 0.5) * (sum2 ** 0.5))


def cosines_calculating(main_dict, word_vec):
    for i in main_dict['pairs']:
        word1 = i[0]
        word2 = i[1]
        cos = cosine(word_vec[word1], word_vec[word2])
        main_dict['cos'].append(cos)
```

Теперь для каждого из датасетов вычисляем коэффициент Спирмена корреляции человеческих оценок близости и косинусов углов между векторами слов. Для этого используем функцию spearmanr, возвращающую два значения: сам коэффициент coef и p-значение.

```
coef, p = spearmanr(d_rel['val'], d_rel['cos'])
print(coef, p)
coef, p = spearmanr(d_sim['val'], d_sim['cos'])
print(coef, p)
```

Получили:
Для датасета wordsim_relatedness
Коэффициент корреляции: 0.737067
p-значение: $1.954179 * 10^{-44}$

Для датасета wordsim_similarity
Коэффициент корреляции: 0.815606
p-значение: $1.192699 * 10^{-49}$

Выводы

Коэффициент корреляции Спирмена варьируется от -1 до 1. Мы получили значения достаточно близкие к 1 (>0.7), из чего можем наблюдать положительную и достаточно сильную связь между совокупностью значений близости, выставленных людьми, и косинусами между векторами из модели fasttest.
p-значение помогает интерпретировать значимость коэффициента, то есть показывает, насколько точна гипотеза о связи между совокупностями значений. Чем меньше p-значение, тем меньше уверенность в результате, в нашем же случае p очень близко к 0, значит, наборы данных действительно коррелированы.