Task 1

2 января 2018 г.

1. Задача 1: сравнение предложений

Дан набор предложений, скопированных с Википедии. Каждое из них имеет "кошачью тему"в одном из трех смыслов:

кошки (животные)

UNIX-утилита cat для вывода содержимого файлов

версии операционной системы OS X, названные в честь семейства кошачьих

Ваша задача — найти два предложения, которые ближе всего по смыслу к расположенному в самой первой строке. В качестве меры близости по смыслу мы будем использовать косинусное расстояние.

Выполните следующие шаги:

1. Скачайте файл с предложениями (sentences.txt).
2. Каждая строка в файле соответствует одному предложению. Считайте их, приведите каждую к нижнему регистру с помощью строковой функции lower().
3. Произведите токенизацию, то есть разбиение текстов на слова. Для этого можно воспользоваться регулярным выражением, которое считает разделителем любой символ, не являющийся буквой: re.split('[^a-z]', t). Не забудьте удалить пустые слова после разделения.
4. Составьте список всех слов, встречающихся в предложениях. Сопоставьте каждому слову индекс от нуля до (d - 1), где d — число различных слов в предложениях. Для этого удобно воспользоваться структурой dict.
5. Создайте матрицу размера n * d, где n — число предложений. Заполните ее: элемент с индексом (i, j) в этой матрице должен быть равен количеству вхождений j-го слова в i-е предложение. У вас должна получиться матрица размера 22 * 254.
6. Найдите косинусное расстояние от предложения в самой первой строке (In comparison to dogs, cats have not undergone...) до всех остальных с помощью функции scipy.spatial.distance.cosine. Какие номера у двух предложений, ближайших к нему по этому расстоянию (строки нумеруются с нуля)? Эти два числа и будут ответами на задание. Само предложение (In comparison to dogs, cats have not undergone...) имеет индекс 0.
7. Запишите полученные числа в файл, разделив пробелом. Обратите внимание, что файл должен состоять из одной строки, в конце которой не должно быть переноса. Пример файла с решением вы можете найти в конце задания (submission-1.txt).
8. Совпадают ли ближайшие два предложения по тематике с первым? Совпадают ли тематики у следующих по близости предложений?

Разумеется, использованный вами метод крайне простой. Например, он не учитывает формы слов (так, cat и cats он считает разными словами, хотя по сути они означают одно и то же), не удаляет из текстов артикли и прочие ненужные слова. Позже мы будем подробно изучать анализ текстов, где выясним, как достичь высокого качества в задаче поиска похожих предложений.

```
In [12]: import re
         import numpy as np
```

```
        import scipy

        s_file = open('sentences.txt')
        n = 0
        for line in s_file:
            n += 1
        s_file = open('sentences.txt')
        sentences = s_file.read().lower()
        s_file.close()
```

In [13]:
```
sentences = re.findall(r'[a-z]+', sentences)
```

In [14]:
```
words = {}
for word in sentences:
    if word in words:
        words[word] += 1
    else:
        words[word] = 1

count = {}
i = 0
for word in words:
    count[word] = i
    i += 1
```

In [15]:
```
d = len(words)
```

In [16]:
```
frec = np.zeros((n, d))

s_file = open('sentences.txt')
s = s_file.read().lower()

i = 0
res = s.split('\n')
for line in res:
    res[i] = re.findall(r'[a-z]+', line)
    i += 1

i = 0
for line in res:
    for word in line:
        frec[i][count[word]] += 1
    i += 1
print frec.shape
s_file.close()
```

```
(22L, 254L)
```

In [17]:
```
import scipy.spatial
dist = []
for i in range(n):
    dist.append(scipy.spatial.distance.cosine(frec[0], frec[i]))
```

In [18]:
```
with open("results.txt", 'w') as file_res:
    final_result = ''
    for elem in dist:
        final_result += str(elem) + " "
    file_res.write(final_result)
```

In [19]:
```
tmp = []
tmp[:] = dist[:]
tmp.pop(0)
ans_1 = dist.index(min(tmp))
tmp.pop(tmp.index(min(tmp)))
ans_2 = dist.index(min(tmp))
```

In [20]:
```
with open("submission-1.txt", 'w') as file_ans:
    s = str(ans_1) + " " + str(ans_2)
    file_ans.write(s)
print s
```

```
6 4
```