Поиск похожих слов и их кластеризация

Зверев Дмитрий, 325-я группа ВМК МГУ

Во время выполнения заданий будем использовать предоставленный на лекции ноутбук *embedding.ipynb*.

Задание 1. Построение списков похожих слов

Будем брать слова из датасета. Реализуем так: с помощью метода `most_similar` модели Word2Vec ищем близкие слова. Используем регулярные выражения и фильтры, чтобы исключить слова с опечатками и т.п. и нормализуем векторы слов. Также дополнительно будет вручную (для большей надежности) проверять получившиеся списки слов на корректность и удалять неподходящие слова:

```
from sklearn.metrics.pairwise import cosine_similarity

#функция для получения ближайших слов
def get_similar_words(word, topn=25, threshold=0.48): #по умолчанию берем 25
    similar_words = wv_embeddings.most_similar(word, topn=topn)
    filtered_words = [w for w, sim in similar_words if sim >= threshold and w.isalpha()]
    embeddings = np.array([wv_embeddings[w] for w in filtered_words])
    embeddings /= np.linalg.norm(embeddings, axis=1, keepdims=True)
    return filtered_words, embeddings\

#построение списка для нашего слова
words, embeddings = get_similar_words('слово')
print("Похожие слова:", words)
```

Отметим, что слова хорошо кластеризуются, если их значения имеют четкие семантические различия и модель эмбеддингов качественно обучена. Это характерно для слов с ярко выраженными значениями, каждое из которых связано с уникальной группой слов. В то же время плохо кластеризуются слова с размытыми значениями, пересекающимися группами слов или редким представлением в данных модели. Несмотря на это, мы будем экспериментировать со словами разной сложности и убедимся, что разделение лучше происходит для слов с четкой разницей между их значениями.

Рассмотрим следующие слова: *“слово”, “звезда”, “машина”, “карта”, “земля”*.

С помощью фрагмента программы выше получаем следующие списки слов:

1) *Для слова “слово”:*
   ['фраза', 'словосочетание', 'словечко', 'произносить', 'высказывание', 'сказать', 'словоформа', 'выговаривать', 'цитировать', 'заверение', 'напутственный', 'пояснять', 'проговорить', 'процитировать', 'мнение', 'обещание', 'подытоживать', 'напутствие', 'резюмировать'];
2) *Для слова “звезда”:*
   ['суперзвезда', 'кинозвезда', 'знаменитость', 'голливуд', 'звездочет', 'небосклон', 'светило', 'эстрада', 'телезвезда', 'болливуд', 'актриса', 'певица', 'селебрити', 'голливудский', 'небосвод', 'болливудский', 'созвездия', 'звездопад', 'шоу'];
3) *Для слова “машина”:*
   ['автомобиль', 'трактор', 'авто', 'стиральная', 'спецмашина', 'мусороуборочный', 'спецавтомобиль', 'посудомоечный', 'стиралка', 'грузовик', 'посудомоечный', 'мотороллер', 'тачка', 'автомоечный', 'снегоочистительный', 'тележка', 'поливомоечный', 'комбайн', 'автофургон', 'мойка'];
4) *Для слова “карта”:*
   ['дебетовый', 'кредитка', 'дисконтный', 'платежный', 'гринкарта', 'симкарта', 'предоплатный', 'чиповый', 'зарплатный', 'картографический', 'банковский', 'кардридер', 'терминал', 'топографический', 'идентификатор', 'картографирование', 'перфокарта', 'медкарта', 'проездной'];
5) *Для слова “земля”:*
   ['пашня', 'угодье', 'плодородный', 'пахотный', 'пастбище', 'сельхозугодье', 'черноземный', 'плодородный', 'землепашец', 'плодородие', 'планета', 'землевладение', 'землянин', 'богарный', 'почва', 'виноградник', 'участок', 'гектар', 'надел', 'пахота', 'грядка'].

Задание 2. Разделить полученный список на кластеры

Для начала используем предложенный алгоритм. Разделим список слов на кластеры, чтобы каждая группа соответствовала своему значению многозначного слова, используя KMeans из библиотеки sklearn. Число кластеров (`n_clusters`) выбирается вручную и может быть оптимизировано, после чего выводятся слова, разбитые по меткам кластеров:

```python
from sklearn.cluster import KMeans

def cluster_words_kmeans(embeddings, n_clusters=3):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    labels = kmeans.fit_predict(embeddings)
    return labels

labels = cluster_words_kmeans(embeddings, n_clusters=4)

for cluster in set(labels):
    print(f"Кластер {cluster}: {[words[i] for i in range(len(labels)) if labels[i] == cluster]}")
```

Для полученных выше списков слов, с учетом удаленных вручную слов и путем подбора оптимального числа кластеров, получим следующие результаты.

1) Для слова *“слово”* (каждый кластер имеет слегка разное значение):
   Кластер 0: ['произносить', 'выговаривать', 'проговорить']
   Кластер 1: ['сказать', 'цитировать', 'пояснять', 'процитировать', 'мнение', 'подытоживать', 'резюмировать']
   Кластер 2: ['фраза', 'словосочетание', 'словечко', 'высказывание', 'словоформа', 'дословно']
   Кластер 3: ['заверение', 'обещание']
   Кластер 4: ['напутственный', 'напутствие']

2) Для слова *“звезда”* (здесь только 2 кластера с предельно понятной разницей):
   Кластер 0: ['суперзвезда', 'кинозвезда', 'знаменитость', 'голливуд', 'эстрада', 'телезвезда', 'болливуд', 'актриса', 'певица', 'селебрити', 'голливудский', 'болливудский', 'шоу']
   Кластер 1: ['звездочет', 'небосклон', 'светило', 'небосвод', 'созвездия', 'звездопад']

3) Для слова *“машина”* (здесь 3 кластера с разновидностями машин, где привычные нам машины соответствуют 1-му кластеру):
   Кластер 0: ['стиральная', 'посудомоечный', 'стиралка', 'посудомоечный', 'автомоечный', 'поливомоечный', 'мойка']
   Кластер 1: ['автомобиль', 'авто', 'спецавтомобиль', 'грузовик', 'автофургон']

Кластер 2: ['трактор', 'спецмашина', 'мусороуборочный', 'мотороллер', 'тачка', 'снегоочистительный', 'тележка', 'комбайн']

4) Для слова *“карта”* (здесь все кластеры имеют отличающиеся значения):
Кластер 0: ['гринкарта', 'симкарта', 'медкарта', 'идентификатор']
Кластер 1: ['кардридер', 'перфокарта']
Кластер 2: ['дебетовый', 'кредитка', 'дисконтный', 'платежный', 'предоплатный', 'чиповый', 'зарплатный', 'банковский', 'терминал', 'проездной']
Кластер 3: ['картографический', 'топографический', 'картографирование']

5) Для слова *“земля”* (здесь все кластеры имеют немного отличающиеся значения):
Кластер 0: ['плодородный', 'черноземный', 'плодородный', 'плодородие', 'богарный', 'почва', 'грядка']
Кластер 1: ['участок', 'надел']
Кластер 2: ['угодье', 'пастбище', 'сельхозугодье', 'землевладение']
Кластер 3: ['пашня', 'пахотный', 'землепашец', 'виноградник', 'гектар', 'пахота']
Кластер 4: ['планета', 'землянин']

Как можно видеть, в целом, слова кластеризуются относительно неплохо и качественно.

Теперь попробуем реализовать собственный алгоритм, идея которого предложена в файле “Практическая работа”, создав 2 кластера.
План следующий:

1) поиск двух самых удаленных слов:
- вычисляется матрица косинусных расстояний между всеми эмбеддингами;
- находятся 2 слова с максимальным расстоянием;

2) выбор центроидов: векторы этих двух слов используются как начальные центроиды;
3) разделение слов на кластеры: каждое слово помещается в тот кластер, к центроиду которого оно ближе.

В результате получаем список слов, разбитый на 2 кластера.

Код будет следующим:

```
from sklearn.metrics.pairwise import cosine_similarity

#матрица кос. расстояний
similarity_matrix = cosine_similarity(embeddings)

#поиск 2 самых удаленных слов (центроидов)
dist_matrix = 1 - similarity_matrix
idx1, idx2 = np.unravel_index(np.argmax(dist_matrix), dist_matrix.shape)
centroid1, centroid2 = embeddings[idx1], embeddings[idx2]

#разделение на 2 кластера
cluster1, cluster2 = [], []
for i, vector in enumerate(embeddings):
    if np.linalg.norm(vector - centroid1) < np.linalg.norm(vector - centroid2):
        cluster1.append(custom_words[i])
    else:
        cluster2.append(custom_words[i])

print("Кластер 1:", cluster1)
print("Кластер 2:", cluster2)
```

Рассмотрим, как данный алгоритм кластеризует похожие на слово *“звезда”* слова (выбрали именно это слово, т.к. оно кластеризуется на 2 группы). Результат будем таким:

Кластер 1: ['суперзвезда', 'кинозвезда', 'знаменитость', 'голливуд', 'эстрада', 'телезвезда', 'болливуд', 'актриса', 'певица', 'селебрити', 'голливудский', 'болливудский']
Кластер 2: ['звездочет', 'небосклон', 'светило', 'небосвод', 'созвездия', 'звездопад', 'шоу']

Результат почти полностью повторяет полученное ранее предложенным другим алгоритмом с тем лишь отличием, что слово “*шоу*” вошло в другой, не соответствующий смыслу этого слова кластер, что может быть связано с чувствительностью выбора центроидов, т.к. слова могут быть редкими или уникальными, и выбор двух самых удаленных слов как центроидов может быть неидеальным.

Задание 3 (домашнее)

Будем искать 2 самых близких по косинусному расстоянию слова в файле с 100 леммами и в файле с 5000 леммами (независимо).

Алгоритм: читаем слова из файла (удаляя дублирующиеся слова), затем вычисляем косинусное расстояние между всеми парами слов и находим пару слов с максимальной схожестью.

Соответствующий код представлен ниже:

```
def find_closest_pair(file_path):
    with open(file_path, "r", encoding="utf-8") as file:
        unique_words = list(set(file.read().splitlines()))

    valid_words = [word for word in unique_words if word in wv_embeddings] #оставляем только те слова, которые есть в модели

    embeddings = np.array([wv_embeddings[word] for word in valid_words])

    #вычисляем кос. расстояние между всеми парами слов
    similarity_matrix = cosine_similarity(embeddings)
    np.fill_diagonal(similarity_matrix, -1)  #не сравниваем слово с самим собой

    idx1, idx2 = np.unravel_index(np.argmax(similarity_matrix), similarity_matrix.shape)
    return valid_words[idx1], valid_words[idx2], similarity_matrix[idx1, idx2]

closest_pair = find_closest_pair("words_big.txt")
print(f"Самые близкие слова: {closest_pair[0]}, {closest_pair[1]}; косинусное расстояние: {closest_pair[2]:.4f}")
```

Так, для файла *words_big.txt* получаем следующий результат:

Самые близкие слова: июль, июнь; косинусное расстояние: 0.9733

Для файла *words_small.txt:*

Самые близкие слова: мальчик, девочка; косинусное расстояние: 0.8899