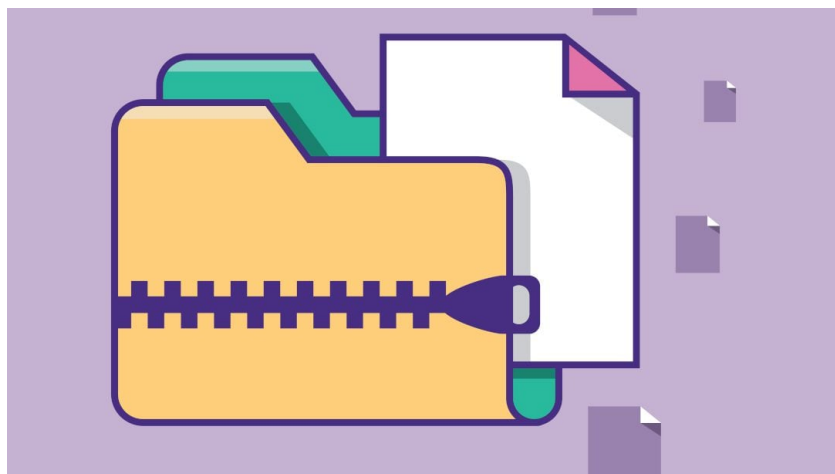


ایندکس معکوس فشرده

Compress Inverted Index



سید محمد حسین هاشمی ۴۰۲۲۳۶۳۱۴۳

فروردین ۱۴۰۳

فهرست مطالب

۲	۱	کلاس Inverted Index
۲	۲	متد -init-
۳	۳	متد read-from-file
۴	۴	متد -compress
۵	۵	متد extract-d-w
۶	۶	متد -decompress
۷	۷	متد read-from-file
۸	۸	متد compress
۹	۹	متد search
۱۰	۱۰	خروجی

۱ کلاس Inverted Index

```
inverted_index.py

1 class InvertedIndex:
2     """ A very simple inverted index. """
```

تمام کدهای مربوطه در این کلاس نوشته می‌شود

۲ متد __init__

```
inverted_index.py

1 def __init__(self):
2     """ Create an empty inverted index. """
3
4     self.postings = None # used for compressed mode
5     self.indexes = None # used for compressed mode
6
7     self.invertedIndex = {}
8     self.compressStatus = False
9     self.matchCount = 1
```

در اینجا یک دیکشنری خالی برای ذخیره ایندکس‌های معکوس، دو متغیر `indexes` و `postings` برای حالت فشرده و همچنین وضعیت فشرده بودن و یا نبودن `compressStatus` ذخیره و در نهایت تعداد کلمات مشابه برای فشرده سازی را در `matchCount` ذخیره می‌کنیم.

۳ متد read-from-file

```
inverted_index.py

1 def read_from_txt(self, file_name):
2     """ Construct index from given file
3
4     >>> ii = InvertedIndex()
5     >>> ii.read_from_txt('Datasets/example.txt')
6     >>> ii.invertedIndex
7     {'first': {1}, 'document': {1, 2, 3}, 'second': {2}, 'third': {3}}
8     """
9     with open(file_name, encoding="utf8") as file:
10         record_id = 0
11         for line in file:
12             record_id += 1
13             for word in re.split('[^a-zA-z]', line):
14                 if len(word) > 0:
15                     word = word.lower()
16                     if word not in self.invertedIndex:
17                         self.invertedIndex[word] = set()
18                     self.invertedIndex[word].add(record_id)
```

در این متد فایل حاوی متون دریافت و خوانده می‌شود و لازم به ذکر است که تمامی متحوا درون یک خط قرار می‌گیرد.

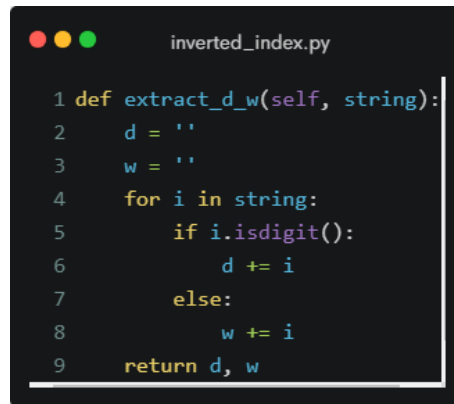
۴ متد compress-

```
inverted_index.py

1 def _compress(self):
2     self.indexes = ''
3     current = '' # current input to check to other
4     index = 0
5     setCurrent = True
6     matchHistory = False
7     words = list(self.invertedIndex.keys())
8     while index < len(words):
9         if words[index] == '': # chk for null
10             index += 1
11             continue
12         if setCurrent: # set input for match to other
13             current = words[index]
14             self.indexes += ',' + str(len(words[index])) + words[index]
15             index += 1
16             setCurrent = False
17             continue
18         ind = -1 # match characters
19         for i in range(min(len(words[index]), len(current))):
20             if current[i] == words[index][i]:
21                 ind = i
22         if ind < self.matchCount: # does not match
23             setCurrent = True
24             matchHistory = False
25         else: # match
26             if matchHistory: # first match
27                 self.indexes += '@'
28             else: # other match
29                 self.indexes += '*'
30                 matchHistory = True
31             self.indexes += str(len(words[index])) + words[index][ind + 1:]
32             index += 1
33     self.indexes = words[1:]
34     self.compressStatus = True
```

در این متد ایندکس‌های معکوس را به روش front-coding فشرده سازی می‌کنیم.

۵ متد extract-d-w



```
1 def extract_d_w(self, string):
2     d = ''
3     w = ''
4     for i in string:
5         if i.isdigit():
6             d += i
7         else:
8             w += i
9     return d, w
```

در این متد کلمه و عدد را در رشته جداسازی می‌کنیم که در متد decopress- استفاده می‌شود.

۶ متد decompress -

```
inverted_index.py

1 def _decompress(self):
2     inp = self.indexes.split(',')
3     output = []
4     for words in inp:
5         if words == "":
6             continue
7         data = words.split('*')
8         if len(data) == 1:
9             output.append(self.extract_d_w(data[0])[1])
10            continue
11        count, current = self.extract_d_w(data[0])
12        output.append(current)
13        for string in data[1].split('@'):
14            d, w = self.extract_d_w(string)
15            output.append(current[: (int(d) - len(w))] + w)
16    return output
```

این متد عکس متد compress عمل کرده و ایندکس‌ها را بصورت یک لیست برمی‌گرداند.

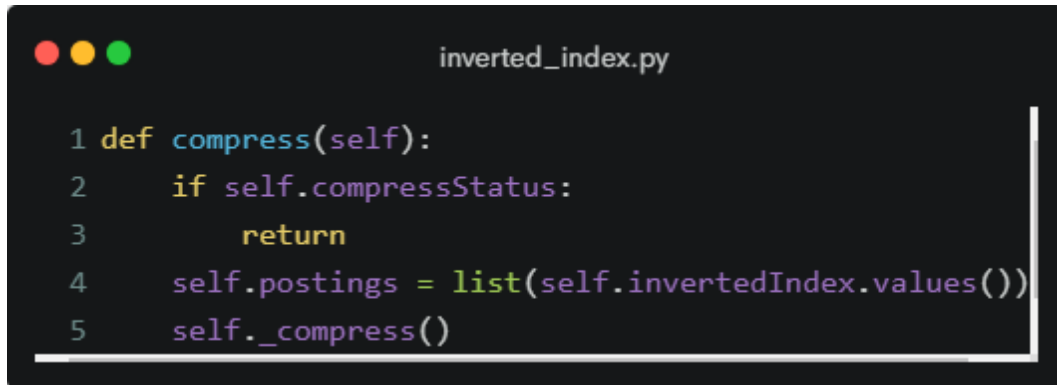
۷ متد read-from-file

```
inverted_index.py

1 def read_from_txt(self, file_name):
2     """ Construct index from given file
3
4     >>> ii = InvertedIndex()
5     >>> ii.read_from_txt('Datasets/example.txt')
6     >>> ii.invertedIndex
7     {'first': {1}, 'document': {1, 2, 3}, 'second': {2}, 'third': {3}}
8     """
9     with open(file_name, encoding="utf8") as file:
10         record_id = 0
11         for line in file:
12             record_id += 1
13             for word in re.split('[^a-zA-z]', line):
14                 if len(word) > 0:
15                     word = word.lower()
16                     if word not in self.invertedIndex:
17                         self.invertedIndex[word] = set()
18                     self.invertedIndex[word].add(record_id)
```

در این متد فایل حاوی متون دریافت و خوانده می‌شود و لازم به ذکر است که تمامی متحوا درون یک خط قرار می‌گیرد.

متد compress ۸



```
1 def compress(self):
2     if self.compressStatus:
3         return
4     self.postings = list(self.invertedIndex.values())
5     self._compress()
```

این متد به قولی handler برای متد compress-است. که لیست postings را جدا می‌کند.

```

inverted_index.py

1 def search(self, search):
2     """ Search with inverted indexes
3
4     >>> ii = InvertedIndex()
5     >>> ii.read_from_txt('Datasets/example.txt')
6     >>> ii.search('first')
7     {1}
8     """
9     search = map(lambda x: x.lower(), re.split('[^a-zA-z]', search))
10
11     results = set()
12
13     if self.compressStatus:
14         for key in search:
15             index = self.indexes.index(key)
16             if index > -1:
17                 if len(results) == 0:
18                     results = set(x for x in self.postings[index])
19                 else:
20                     results.intersection_update(self.postings[index])
21     else:
22         available_keys = self.invertedIndex.keys()
23         for key in search:
24             if key in available_keys:
25                 if len(results) == 0:
26                     results = set(x for x in self.invertedIndex[key])
27                 else:
28                     results.intersection_update(self.invertedIndex[key])
29
30     return results

```

در این متد عملیات جست‌وجو انجام می‌شود اما این جست‌وجو در دو حالت فشرده نشده و فشرده شده انجام می‌شود که هدف آن مقایسه دو حالت فشرده شده و غیر فشرده است.

۱۰ خروجی

```
inverted_index.py

1 if __name__ == '__main__':
2     if len(sys.argv) != 2:
3         print("Usage: python inverted_index.py [file_name]")
4         sys.exit(1)
5
6     filename = sys.argv[1]
7
8     ii = InvertedIndex()
9     ii.read_from_txt(filename)
10
11     print('Search [0]')
12     print('See inverted index [1]')
13     print('compress indexes [2]')
14     print('See compressed indexes [3]')
15     print('See uncompressed indexes [4]')
16     status = input('Choose action or any key to exit: ')
17     while status.isdigit() and 0 <= int(status) <= 4:
18         if status == '0':
19             for doc in ii.search(input('search: ')):
20                 print(doc)
21         elif status == '1':
22             print('inverted index')
23             print('input'.ljust(30), 'repeats'.ljust(10), 'documents')
24             for word, indexes in ii.invertedIndex.items():
25                 print(word.ljust(30), str(len(indexes)).ljust(10), indexes)
26         elif status == '2':
27             ii.compress()
28             print('Successfully compressed')
29         elif status == '3':
30             print(ii.get_compress_index())
31         else:
32             print(ii.get_decompress_index())
33
34     status = input('Choose action or any key to exit: ')
35     print('Bye')
```

همانطور که مشاهده می‌شود ترمینال ساده‌ای جهت کارکردن با برنامه ایجاد شده است. نکته مهم این است که برای دیدن نسخه فشرده ابتدا باید فرمان فشرده سازی اعمال شود و همچنین با این کار جست‌وجو با استفاده از ایندکس‌های فشرده انجام می‌شود.