```
print(forward_segment('就读北京大学', dic))
print(forward_segment('研究生命起源', dic))
```

输出：

```
['就读', '北京大学']
['研究生', '命', '起源']
```

第二句话就会产生误差了，我们是需要把“研究”提取出来，结果按照正向最长匹配算法就提取出了“研究生”，所以人们就想出了逆向最长匹配。

3. 逆向最长匹配

```
def backward_segment(text, dic):
    word_list = []
    i = len(text) - 1
    while i >= 0:                                # 扫描位置作为终点
        longest_word = text[i]                   # 扫描位置的单字
        for j in range(0, i):                    # 遍历[0, i]区间作为待查询词
            word = text[j: i + 1]                # 取出[j, i]区间作为待查询单
            if word in dic:
                if len(word) > len(longest_word):    # 越长优先级越高
                    longest_word = word
                    break
        word_list.insert(0, longest_word)        # 逆向扫描，所以越先查出的单词
        i -= len(longest_word)
    return word_list

dic = load_dictionary()
print(backward_segment('研究生命起源', dic))
print(backward_segment('项目的研究', dic))
```

输出：

```
['研究', '生命', '起源']
['项', '目的', '研究']
```

第一句正确了，但下一句又出错了，可谓拆东墙补西墙。另一些人提出综合两种规则，期待它们取长补短，称为双向最长匹配。

2.6 HanLP的词典分词实现

1. **DoubleArrayTrieSegment**

   DoubleArrayTrieSegment分词器是对DAT最长匹配的封装，默认加载hanlp.properties中CoreDictionaryPath制定的词典。

2. **去掉停用词**

   停用词词典文件：data/dictionnary/stopwords.txt

   该词典收录了常见的中英文无意义词汇(不含敏感词)，每行一个词。