734. Sentence Similarity

Nov. 25, 2017 | 14.7K views

Average Rating: 3.20 (30 votes)

Given two sentences `words1, words2` (each represented as an array of strings), and a list of similar word pairs `pairs`, determine if two sentences are similar.

For example, "great acting skills" and "fine drama talent" are similar, if the similar word pairs are `pairs = [["great", "fine"], ["acting","drama"], ["skills","talent"]]`.

Note that the similarity relation is not transitive. For example, if "great" and "fine" are similar, and "fine" and "good" are similar, "great" and "good" are **not** necessarily similar.

However, similarity is symmetric. For example, "great" and "fine" being similar is the same as "fine" and "great" being similar.

Also, a word is always similar with itself. For example, the sentences `words1 = ["great"], words2 = ["great"], pairs = []` are similar, even though there are no specified similar word pairs.

Finally, sentences can only be similar if they have the same number of words. So a sentence like `words1 = ["great"]` can never be similar to `words2 = ["doubleplus","good"]`.

Note:

- The length of `words1` and `words2` will not exceed `1000`.
- The length of `pairs` will not exceed `2000`.
- The length of each `pairs[i]` will be `2`.
- The length of each `words[i]` and `pairs[i][j]` will be in the range `[1, 20]`.

Approach #1: Set [Accepted]

Intuition and Algorithm

To check whether `words1[i]` and `words2[i]` are similar, either they are the same word, or `(words1[i], words2[i])` or `(words2[i], words1[i])` appear in `pairs`.

To check whether `(words1[i], words2[i])` appears in `pairs` quickly, we could put all such pairs into a Set structure.

Java | Python — Copy

```python
class Solution(object):
    def areSentencesSimilar(self, words1, words2, pairs):
        if len(words1) != len(words2): return False

        pairset = set(map(tuple, pairs))
        return all(w1 == w2 or (w1, w2) in pairset or (w2, w1) in pairset
                   for w1, w2 in zip(words1, words2))
```

Complexity Analysis

- Time Complexity: $O(N + P)$, where N is the maximum length of `words1` and `words2`, and P is the length of `pairs`.
- Space Complexity: $O(P)$, the size of `pairs`. Intermediate objects created in evaluating whether a pair of words are similar are created one at a time, so they don't take additional space.

Analysis written by: @awice.

Rate this article:

Previous Next

Comments: 13

Sort By

Type comment here... (Markdown is supported)

Preview Post

zhangyan985211 ★ 95 April 3, 2018 11:34 AM

I think we can use Map<String, Set instead of only set , that's because the word may contain some characters as # which will cause an error by using set.contains() method.

29 Share Reply

netsim ★ 12 February 21, 2018 3:53 AM

This solution seems assume it only needs to check the word similarity at the same position. But the question does not seem have that assumption ?

12 Share Reply

SHOW 2 REPLIES

digital_nomad ★ 3 August 24, 2018 4:40 PM

Isn't the time complexity of `equals` O(N) for a String? I believe the time complexity analysis is wrong here.

3 Share Reply

SHOW 2 REPLIES

matt316 ★ 2 January 25, 2019 11:05 AM

I think they should consider this test case also..for the given tc...code is failing but I think it should give true.

["skill","acting","great"]
["fine","drama","talent"]

Read More

1 Share Reply

SHOW 3 REPLIES

genius1wjc ★ 4 December 17, 2017 12:00 AM

Why use `contains`? Can we just use `equals`? If we use `contains`, something like "asd#cvb".contains("sd#cv") will return true, but "sd" and "cv" are not similar in this case

1 Share Reply

SHOW 2 REPLIES

SherMM ★ 50 February 11, 2018 9:11 PM

I don't think these solutions are very flexible. It assumes an ordering of words1 and words2, as you can see by the second for loop in the java solution and the zip() usage in the Python solution. Coudn't the sentence be "similar" even if the words that are similar don't line up exactly with each other?

0 Share Reply

lch04 ★ 8 January 16, 2018 1:22 PM

@genius1wjc what do you mean? The code does full string comparison, not substring.

0 Share Reply

SHOW 1 REPLY

sschangi ★ 183 November 27, 2017 11:57 AM

@awice I think you forgot to change O(N) to O(P) in the space complexity analysis.

0 Share Reply

ManuelP ★ 752 November 27, 2017 5:01 AM

(Edit: got fixed) Isn't space complexity O(P)? Why do you add N? Those intermediate objects exist only one at a time, not all at the same time. So they can occupy the same space over and over again.

0 Share Reply

SHOW 2 REPLIES

ManuelP ★ 752 November 26, 2017 8:35 PM

No need for such optimization, since `words1` and `words2` are never longer than 56 (despite 1000 being allowed) and `pairs` is never longer than 73 (despite 2000 being allowed).

0 Share Reply