734. Sentence Similarity (/problems/sentence-similarity/)

734. Sentence Similarity

/?return=/articles/sentence-similarity/) (/ratings/107/300/?return=/articles/sentence-similarity/) (/ratings/107/300/?return=/articles/sentence-similarity/)

Average Rating: 4.50 (4 votes)

Nov. 25, 2017 | 2.5K views

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

Articles >

```
class Solution {
    public boolean areSentencesSimilar(
            String[] words1, String[] words2, String[][] pairs) {
        if (words1.length != words2.length) return false;

        Set<String> pairset = new HashSet();
        for (String[] pair: pairs)
            pairset.add(pair[0] + "#" + pair[1]);

        for (int i = 0; i < words1.length; ++i) {
            if (!words1[i].equals(words2[i]) &&
                    !pairset.contains(words1[i] + "#" + words2[i]) &&
                    !pairset.contains(words2[i] + "#" + words1[i]))
                return false;
        }
        return true;
    }
}
```

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 (https://leetcode.com/awice).

Rate this article:

(/ratings/107/300/?return=/articles/sentence-similarity/) (/ratings/107/300/?return=/articles/sente

Previous (/articles/my-calendar-ii/) Next (/articles/sentence-similarity-ii/)

Join the conversation

Login to Reply

N (https://discuss.leetcode.com/user/netsim)

netsim commented 2 weeks ago

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 ?

S (https://discuss.leetcode.com/user/shermm)

SherMM commented last month

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?

G (https://discuss.leetcode.com/user/genius1wjc)

genius1wjc commented last month

@lch04 (https://discuss.leetcode.com/uid/2645) I couldn't view the question and the solution anymore. As you said, it should do full string comparison, not substring. That's why we should just use `equals` rather than `contains`. Or maybe the solution has been edited to use `equals` now

Articles >

lch04 commented last month

(https://discuss.leetcode.com/user/lch04)

@genius1wjc (https://discuss.leetcode.com/uid/40082) what do you mean? The code does full string comparison, not substring.

genius1wjc commented 3 months ago

(https://discuss.leetcode.com/user/genius1wjc)

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

sschangi commented 3 months ago

(https://discuss.leetcode.com/user/sschangi)

@awice (https://discuss.leetcode.com/uid/71269) I think you forgot to change O(N) to O(P) in the space complexity analysis.

awice commented 3 months ago

(https://discuss.leetcode.com/user/awice)

Corrected, thanks.

ManuelP commented 3 months ago

(https://discuss.leetcode.com/user/manuelp)

(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.

ManuelP commented 3 months ago

(https://discuss.leetcode.com/user/manuelp)

No need (https://discuss.leetcode.com/topic/112149/1-liner) 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).

View original thread (https://discuss.leetcode.com/topic/112016)