Daily Coding Problem #153

Problem

Find an efficient algorithm to find the smallest distance (measured in number of words) between any two given words in a string.

For example, given words "hello", and "world" and a text content of "dog cat hello cat dog dog hello cat world", return 1 because there's only one word "cat" in between the two words.

Solution

We can translate this problem into a more algorithmic format. We can first find all the indices of `word0` and `word1` in `text`. Then with the two indices lists, we have the problem of finding a number from each of the lists that minimizes their difference.

For example, given `[1, 10, 33]` and `[5, 6, 15, 32]`, the two numbers would be 33 and 32.

To solve this problem, we can use a greedy strategy. We keep two pointers `i`, for the `word0` indices, and `j`, for the `word1` indices. Then we explore different `i` and `j` while keeping the minimum distance we've seen of their values.

For example, this is the initial state:

[1, 10, 33] ^ i

[5, 6, 15, 32] ^ j

And the minimum distance would begin with `abs(5 - 1) == 4`. Then we iterate until `i` or `j` is out of index: if the value indexed by `i`, `word0_indices[i]`, is lower than at `j`, we increment `i`, and otherwise increment `j`.

This process must work since the optimal solution must make the value of the lower number higher in order to minimize the difference.

```
def min_distance(text, word0, word1):
    text_words = [w.strip() for w in text.split(' ')]
    print text_words

    word0_indices = [i for i, w in enumerate(text_words) if w == word0]
    word1_indices = [i for i, w in enumerate(text_words) if w == word1]

    if not word0_indices or not word1_indices: # one of the words doesn't exist.
        return float('inf')

    i = j = 0

    min_distance = abs(word0_indices[i] - word1_indices[j])

    while i < len(word0_indices) and j < len(word1_indices):

        current_distance = abs(word0_indices[i] - word1_indices[j])
        min_distance = min(min_distance, current_distance)

        if word0_indices[i] < word1_indices[j]:
            i += 1
        else:
            j += 1
    return min_distance - 1 # Don't count the last step to get to word1
```

This takes O(n) space and time.