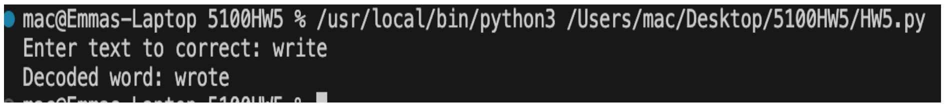
Foundations of AI
Yunyu Guo
Assignment 5

This assignment is to write a spelling fixer using a Hidden Markov model. You will take user input and correct the spelling using the Viterbi algorithm

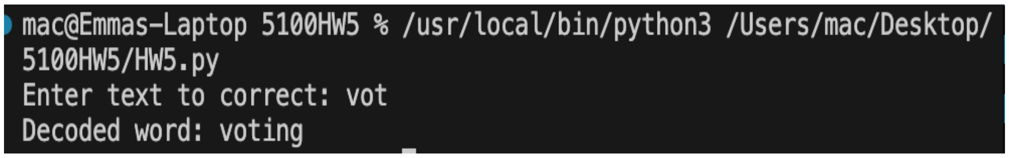
- Questions specific to this assignment:
  - Please give an example of a word which was correctly spelled by the user, but which was incorrectly "corrected" by the algorithm. Why did this happen?



```
mac@Emmas-Laptop 5100HW5 % /usr/local/bin/python3 /Users/mac/Desktop/5100HW5/HW5.py
Enter text to correct: write
Decoded word: wrote
```

In this example, I typed "write" but was incorrectly corrected to "wrote". This is because in the dictionary, there is only correct word "wrote" and "write" does not exist in the correct words list.

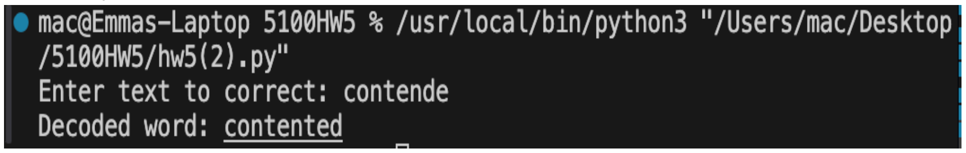
  - Please give an example of a word which was incorrectly spelled by the user, but which was still incorrectly "corrected" by the algorithm. Why did this happen?



```
mac@Emmas-Laptop 5100HW5 % /usr/local/bin/python3 /Users/mac/Desktop/
5100HW5/HW5.py
Enter text to correct: vot
Decoded word: voting
```

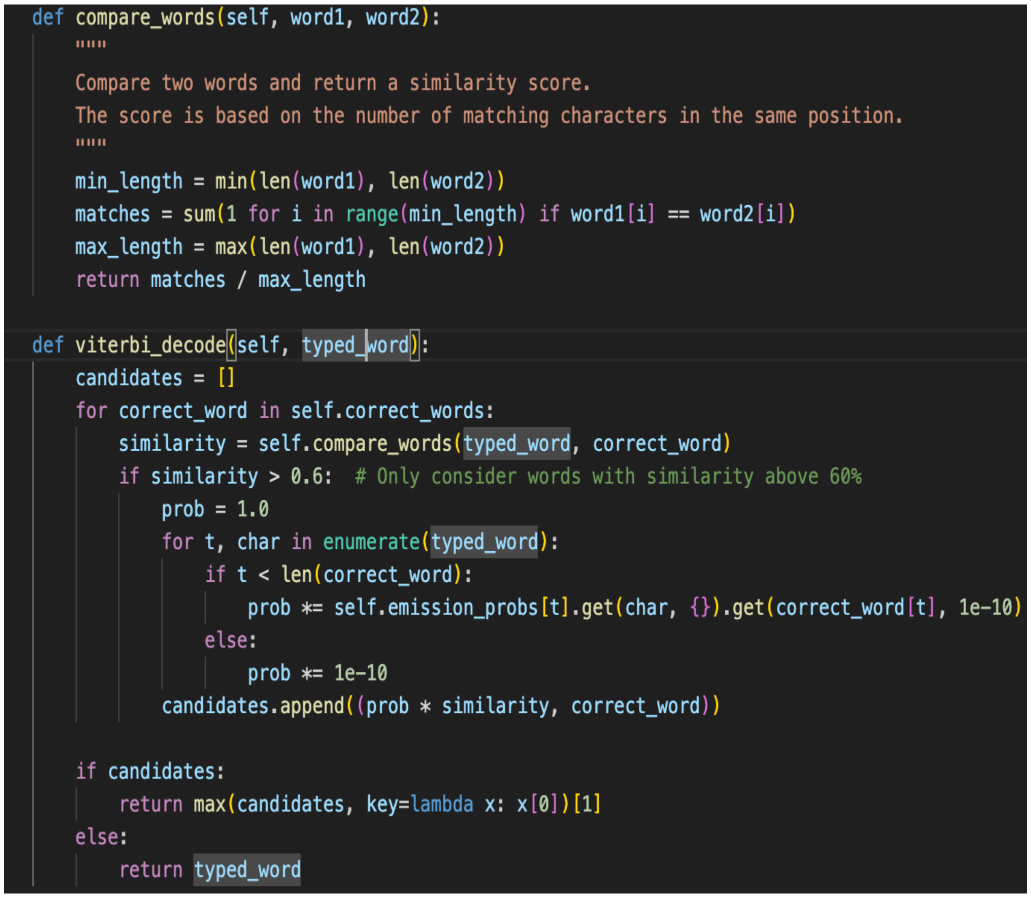
It this example, I typed "vot" (correct word should be "vote", but was incorrectly corrected to "voting". This is because in the dictionary, only "voting" in the correct words list, the algorithm does not capture the path to "vote".

  - Please give an example of a word which was incorrectly spelled by the user, and was correctly corrected by the algorithm. Why was this one correctly corrected, while the previous two were not?



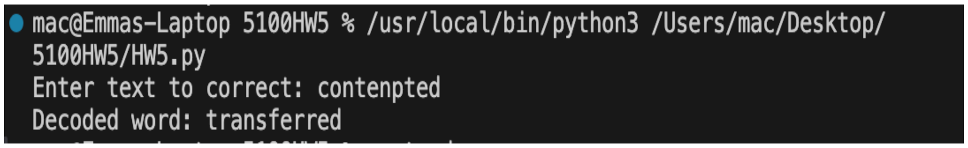
```
mac@Emmas-Laptop 5100HW5 % /usr/local/bin/python3 "/Users/mac/Desktop
/5100HW5/hw5(2).py"
Enter text to correct: contende
Decoded word: contented
```

In this example, the incorrectly spelled word "contende" is in the dictionary, it was correctly corrected to "contented", which also in the correct words list. The reason why this one was correctly corrected is because both incorrect and correct words are in the misspelling and correct words lists. Also I implement a comparison to compare similarity of 2 words, get a score, then multiply emission probabilities to select the best correction.

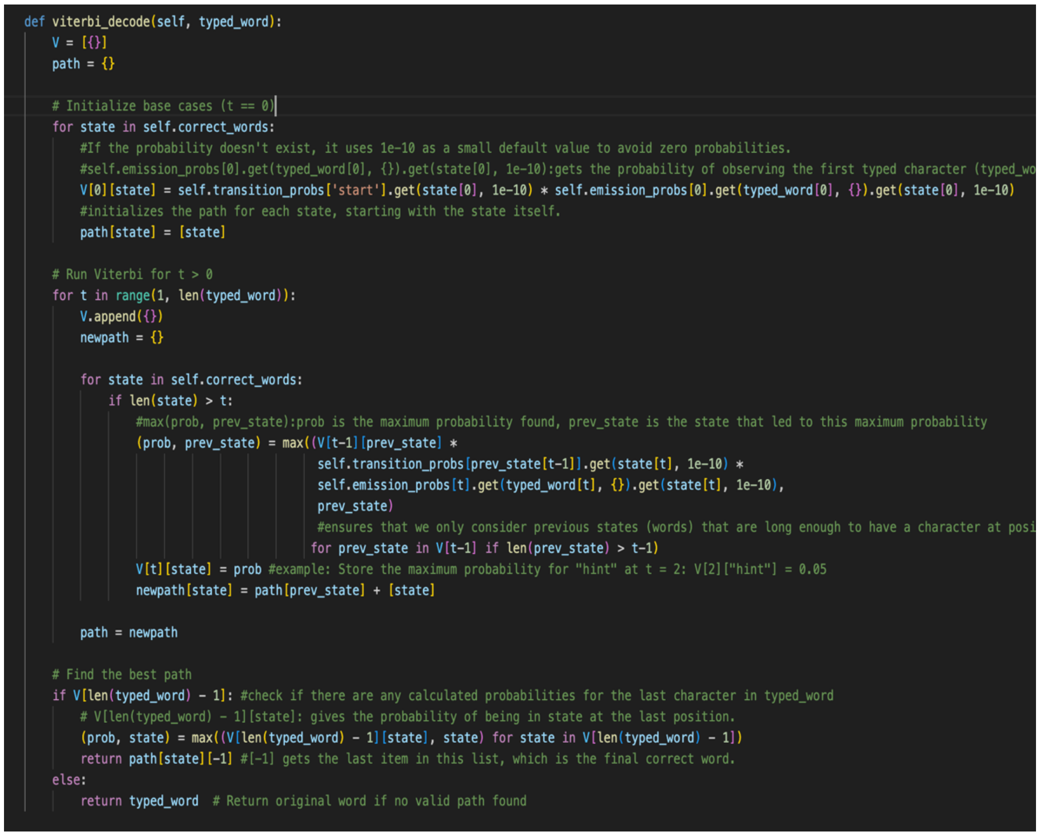


```python
def compare_words(self, word1, word2):
    """
    Compare two words and return a similarity score.
    The score is based on the number of matching characters in the same position.
    """
    min_length = min(len(word1), len(word2))
    matches = sum(1 for i in range(min_length) if word1[i] == word2[i])
    max_length = max(len(word1), len(word2))
    return matches / max_length

def viterbi_decode(self, typed_word):
    candidates = []
    for correct_word in self.correct_words:
        similarity = self.compare_words(typed_word, correct_word)
        if similarity > 0.6:  # Only consider words with similarity above 60%
            prob = 1.0
            for t, char in enumerate(typed_word):
                if t < len(correct_word):
                    prob *= self.emission_probs[t].get(char, {}).get(correct_word[t], 1e-10)
                else:
                    prob *= 1e-10
            candidates.append((prob * similarity, correct_word))

    if candidates:
        return max(candidates, key=lambda x: x[0])[1]
    else:
        return typed_word
```

While using the Viterbi algorithm did not correct the word. In the following example, the incorrect word “contenpted” was decoded to “transferred”.



```
● mac@Emmas-Laptop 5100HW5 % /usr/local/bin/python3 /Users/mac/Desktop/
  5100HW5/HW5.py
  Enter text to correct: contenpted
  Decoded word: transferred
```



```python
def viterbi_decode(self, typed_word):
    V = [{}]
    path = {}

    # Initialize base cases (t == 0)
    for state in self.correct_words:
        #If the probability doesn't exist, it uses 1e-10 as a small default value to avoid zero probabilities.
        #self.emission_probs[0].get(typed_word[0], {}).get(state[0], 1e-10):gets the probability of observing the first typed character (typed_wo
        V[0][state] = self.transition_probs['start'].get(state[0], 1e-10) * self.emission_probs[0].get(typed_word[0], {}).get(state[0], 1e-10)
        #initializes the path for each state, starting with the state itself.
        path[state] = [state]

    # Run Viterbi for t > 0
    for t in range(1, len(typed_word)):
        V.append({})
        newpath = {}

        for state in self.correct_words:
            if len(state) > t:
                #max(prob, prev_state):prob is the maximum probability found, prev_state is the state that led to this maximum probability
                (prob, prev_state) = max((V[t-1][prev_state] *
                                          self.transition_probs[prev_state[t-1]].get(state[t], 1e-10) *
                                          self.emission_probs[t].get(typed_word[t], {}).get(state[t], 1e-10),
                                          prev_state)
                                          #ensures that we only consider previous states (words) that are long enough to have a character at posi
                                         for prev_state in V[t-1] if len(prev_state) > t-1)
                V[t][state] = prob #example: Store the maximum probability for "hint" at t = 2: V[2]["hint"] = 0.05
                newpath[state] = path[prev_state] + [state]

        path = newpath

    # Find the best path
    if V[len(typed_word) - 1]: #check if there are any calculated probabilities for the last character in typed_word
        # V[len(typed_word) - 1][state]: gives the probability of being in state at the last position.
        (prob, state) = max((V[len(typed_word) - 1][state], state) for state in V[len(typed_word) - 1])
        return path[state][-1] #[-1] gets the last item in this list, which is the final correct word.
    else:
        return typed_word  # Return original word if no valid path found
```

- The usual questions:
  - How long did this assignment take you? (1 sentence)
    3 days

  - Whom did you work with, and how? (1 sentence each)
    - Discussing the assignment with others is encouraged, as long as you don't share the code.

    I discussed with a student about how to calculate elements in transition matrix, emission matrix.

  - Which resources did you use? (1 sentence each)
    - For each, please list the URL and a brief description of how it was useful.
      I used the spell-testset1.txt to train model:
      https://www.kaggle.com/datasets/bittlingmayer/spelling

      Viterbi algorithm: https://www.geeksforgeeks.org/viterbi-algorithm-for-hidden-markov-models-hmms/
  - A few sentences about:
    - What was the most difficult part of the assignment?

How to calculate the emission probability, as it represents the likelihood of observing a particular output given a hidden state (correct word). Also how to calculate the Viterbi.

- What was the most rewarding part of the assignment?
  Practice on how word correction works

- What did you learn doing the assignment?
  What is Viterbi algorithm and it might not properly work on correcting word.

- Constructive and actionable suggestions for improving assignments, office hours, and class time are always welcome.
  It would be great to have office hours before class, maybe consider virtual and in-person.