```
def read_file(input_file):
    """Read content from a file."""
    try:
        with open(input_file, "r", encoding="utf-8") as file:
            return file.read()
    except FileNotFoundError:
        raise FileNotFoundError(f"The file '{input_file}' was not found.")

def write_file(output_file, content):
    """Write content to a file."""
    with open(output_file, "w", encoding="utf-8") as file:
        file.write(content)
    print(f"Content successfully saved to {output_file}")



def load_dictionary(file_path):
    """Load words from a dictionary file."""
    with open(file_path, 'r', encoding='utf-8') as file:
        return set(line.strip() for line in file)



from collections import Counter

def word_length_similarity(misspelt_word, candidate_word):
    """Binary score: 1.0 if lengths match, 0.0 otherwise."""
    return 1.0 if len(misspelt_word) == len(candidate_word) else 0.0

def character_frequency_similarity(misspelt_word, candidate_word):
    """Compute similarity based on character frequency."""
    misspelt_counter = Counter(misspelt_word)
    candidate_counter = Counter(candidate_word)
    matching_count = sum(min(misspelt_counter[char], candidate_counter[char]) for char in misspelt_counter)
    return matching_count / len(misspelt_word)

def position_similarity(misspelt_word, candidate_word):
    """Count matching characters at the same positions."""
    match_count = sum(1 for m_char, c_char in zip(misspelt_word, candidate_word) if m_char == c_char)
    return match_count / min(len(misspelt_word), len(candidate_word))

def first_letter_similarity(misspelt_word, candidate_word):
    """Check if the first letters of both words match."""
    return 1.0 if misspelt_word[0] == candidate_word[0] else 0.0



def levenshtein_distance(word1, word2):
    """Compute the Levenshtein distance between two words."""
    len_word1, len_word2 = len(word1), len(word2)
    matrix = [[0] * (len_word2 + 1) for _ in range(len_word1 + 1)]

    for i in range(len_word1 + 1):
        matrix[i][0] = i
    for j in range(len_word2 + 1):
        matrix[0][j] = j

    for i in range(1, len_word1 + 1):
        for j in range(1, len_word2 + 1):
            cost = 0 if word1[i-1] == word2[j-1] else 1
            matrix[i][j] = min(
                matrix[i-1][j] + 1,  # Deletion
                matrix[i][j-1] + 1,  # Insertion
                matrix[i-1][j-1] + cost  # Substitution
            )

    return matrix[len_word1][len_word2]

def distance_similarity(misspelt_word, candidate_word):
    """Levenshtein distance normalized by maximum possible distance."""
    max_distance = max(len(misspelt_word), len(candidate_word))
    lev_distance = levenshtein_distance(misspelt_word, candidate_word)
    return 1 - (lev_distance / max_distance)
```

```
def calculate_similarity_score(misspelt_word, candidate_word):
    """Calculate weighted similarity score between two words."""
    # Weights
    weights = {
        "length": 0.2,
        "frequency": 0.2,
        "position": 0.2,
        "first_letter": 0.2,
        "distance": 0.2
    }

    # Compute heuristic scores
    scores = {
        "length": word_length_similarity(misspelt_word, candidate_word),
        "frequency": character_frequency_similarity(misspelt_word, candidate_word),
        "position": position_similarity(misspelt_word, candidate_word),
        "first_letter": first_letter_similarity(misspelt_word, candidate_word),
        "distance": distance_similarity(misspelt_word, candidate_word)
    }

    # Weighted sum of the scores
    return sum(weights[key] * scores[key] for key in weights)



def suggest_corrections(misspelt_word, dictionary, threshold=0.5):
    """Suggest corrections for a misspelt word."""
    scored_candidates = [
        (candidate, calculate_similarity_score(misspelt_word, candidate))
        for candidate in dictionary
    ]
    # Filter candidates above the threshold
    scored_candidates = [(word, score) for word, score in scored_candidates if score > threshold]
    # Sort by similarity score in descending order
    return sorted(scored_candidates, key=lambda x: x[1], reverse=True)



def process_sentence(sentence, dictionary):
    """
    Process a sentence to find corrections for each word and generate the corrected sentence.

    Parameters:
    - sentence: str, the input sentence to be corrected.
    - dictionary: set, a set of valid Tamil words.

    Returns:
    - corrections: dict, mapping of each word to its suggestions or status.
    - corrected_sentence: str, the sentence after applying corrections.
    """
    words = sentence.split()
    corrections = {}
    corrected_words = []

    for word in words:
        suggestions = suggest_corrections(word, dictionary)
        if not suggestions:
            corrections[word] = "No suggestions"
            corrected_words.append(word)
        elif suggestions[0][1] == 1.0:  # Check for a perfect match
            corrections[word] = "Correct"
            corrected_words.append(word)
        else:
            corrections[word] = [s[0] for s in suggestions]
            corrected_words.append(suggestions[0][0])  # Pick the best suggestion

    corrected_sentence = " ".join(corrected_words)
    return corrections, corrected_sentence



# Example usage
dictionary_file = '/content/tamil_words.txt'  # Replace with your Tamil words file
sentence = 'நான் பூங்க பார்ப்பேன்'  # Replace with your Tamil sentence

# Load dictionary
dictionary = load_dictionary(dictionary_file)
```

```
# Process the sentence
corrections, corrected_sentence = process_sentence(sentence, dictionary)

# Print corrections for the sentence
for word, suggestion in corrections.items():
    if suggestion == "Correct":
        print(f"'{word}': Correct")
    elif suggestion == "No suggestions":
        print(f"'{word}': No suggestions found")
    else:
        print(f"'{word}': Suggestions: {', '.join(suggestion)}")

# Print the corrected sentence
print("Corrected Sentence:", corrected_sentence)
```

```
'நான்': Correct
'பூங்க': Suggestions: பூங்கா, பூங்கு, பரக்க, பூங்கணை, பூங்கோரை, பூங்கலன், பூங்கழல், பூங்கொடி, பூங்காவி, பூங்காடு, பூங்கா
'பார்ப்பேன்': Correct
Corrected Sentence: நான் பூங்கா பார்ப்பேன்
```

```
ntent/tamil_words.txt'  # Replace with your Tamil words file
ுது பாலை குடித்தான். அவனுக்கு புத்தகம் புடிச்சு. அவள் பாடம் நன்றாகப் படித்தார். அவர்கள் கல் அடிக்கிறான். கடைசி நேரத்தில், வேல
```

```
ionary(dictionary_file)
```

```
_sentence = process_sentence(sentence, dictionary)

r the sentence
n corrections.items():
Correct":
}': Correct")
 "No suggestions":
}': No suggestions found")

}': Suggestions: {', '.join(suggestion)}")

sentence
nce:", corrected_sentence)
```

```
'நான்': Correct
'நேத்து': Suggestions: நெத்து, நீத்து, நேற்று, நாத்தி, நேக்கு, நொய்து, நடத்தை, நித்தை, நேர்பு, நொந்து, நேர்வு, நீந்து, நிந்து, ந
'பாலை': Suggestions: பாலி, பாஷை, பாறை, பாரை, பானை, பாளை, பாடை, பாமை, பனலை, புலை, பாதை, பசலை, பாவை,
'குடித்தான்.': Suggestions: குழித்தாமரை, கொடித்தண்டு, குணபத்திரன், குடிக்காணம், குடிக்காவல், காணித்தாயம், குளிர்பானம்,
'அவனுக்கு': Suggestions: அவலரக்கு, அபவாக்கு, அவலுப்பு, அணுக்கரு, அவக்குறி, அகலாங்கு, அலகழங்கு, அலகுகூறு, அலவுற்று,
'புத்தகம்': Correct
'புடிச்சு.': Suggestions: புளிச்சல், புரைச்சல், பிரிச்சல், பொலிச்சல், பிடிசுவர், புட்பாசனி, புரிசாலம், பிடிசாவல், புட்பரசம், பொடிச
'அவள்': Correct
'பாடம்': Correct
'நன்றாகப்': Suggestions: நன்முகம், நற்பாடம், நற்காலம், நற்காமம், நுதற்கண், நன்னிறம், நட்பாளர், நட்பாடல், நீலநாகம், நாகர
'படித்தார்.': Suggestions: படித்திரம், பிரத்தாரம், பவித்திரன், பவித்திரம், பலித்தாலம், படிக்காரம், படிக்காரன், படிமத்தான், படிம
'அவர்கள்': Suggestions: அவரோகம், அவற்கம், அவராகம், அளக்கர், அரங்கன், அதர்வம், அரக்கன், அளக்கல், அரங்கம், அரக்கம்,
'கல்': Correct
'அடிக்கிறான்.': Suggestions: அபிக்கியாதம், அரிக்கரியார், அபிக்கிரகணம், அறிக்கையிடல், அடுக்கேற்றம், அக்கினிமாடன், அநுக்
'கடைசி': Correct
'நேரத்தில்,': Suggestions: நிரத்துதல், நேபத்தியம், நவரத்தினம், நிலத்தியல், நமித்திரர், நிரஸித்தல், நிழத்துதல், நரேந்திரன், நிலத்
'வேலை': Suggestions: வேவை, வயலை, வழலை, விலை, வாலை, வேசை, வேதை, வேடை, வசலை, வலவை, வலசை, வேசா, வ
'முடிக்கிறேன்.': Suggestions: மிடாக்குடியன், முட்டிக்காலன், முடித்தகேள்வி, முலைக்கிரந்தி, மொடாக்குடியன், முறுக்காறுதல், மாவ
Corrected Sentence: நான் நெத்து பாலி குழித்தாமரை அவலரக்கு புத்தகம் புளிச்சல் அவள் பாடம் நன்முகம் படித்திரம் அவரோகம் கல்
```

Start coding or generate with AI.