```
import re
from collections import Counter

#Corpus
# corpus_text = """
# Neural networks are trained using large datasets to recognize patterns and make predictions.
# """
corpus_text = """
I am Vedant Deore studying in Vishwakarma Institute of Technology Pune and I am Third Year Artificial Intelligence and Data Science student
"""

def build_corpus(corpus_text):
    words = re.findall(r'\w+', corpus_text.lower())
    return Counter(words)

#Probability
def probability(word, corpus_counter, total_words):
    return corpus_counter[word] / total_words

def edits1(word):
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(word[:i], word[i:]) for i in range(len(word) + 1)]
    deletes    = [L + R[1:] for L, R in splits if R]
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R) > 1]
    replaces   = [L + c + R[1:] for L, R in splits if R for c in letters]
    inserts    = [L + c + R for L, R in splits for c in letters]
    return set(deletes + transposes + replaces + inserts)

def candidates(word, corpus_counter):
    known_words = known([word], corpus_counter)
    known_edits1 = known(edits1(word), corpus_counter)
    known_edits2 = known([e2 for e1 in edits1(word) for e2 in edits1(e1)], corpus_counter)
    return known_words or known_edits1 or known_edits2 or [word]

def known(words, corpus_counter):
    return set(w for w in words if w in corpus_counter)


def correct(word, corpus_counter, total_words):
    candidates_list = candidates(word, corpus_counter)
    return max(candidates_list, key=lambda w: probability(w, corpus_counter, total_words))

corpus_counter = build_corpus(corpus_text)
total_words = sum(corpus_counter.values())

input_word1 = 'Devant'  # misspelled word
corrected_word1 = correct(input_word1, corpus_counter, total_words)
print(corrected_word1)

input_word2 = "Articicial"  # misspelled word
corrected_word2 = correct(input_word2, corpus_counter, total_words)
print(corrected_word2)

input_word3 = "Vishakarma"  # misspelled word
corrected_word3 = correct(input_word3, corpus_counter, total_words)
print(corrected_word3)


# Print all possible edits
print("\nPossible Edits:")
print(f"All edits 1 away: {edits_info['all_edits1']}")
print(f"Known edits 1 away: {edits_info['known_edits1']}")
print(f"All edits 2 away: {edits_info['all_edits2']}")
print(f"Known edits 2 away: {edits_info['known_edits2']}")
```

```
vedant
artificial
vishwakarma
```

```
import re
from collections import Counter

def build_corpus(corpus_text):
    """
    Build a word frequency counter from the corpus text
```

```
    Build a word frequency counter from the corpus text.
    """
    words = re.findall(r'\w+', corpus_text.lower())
    return Counter(words)

def probability(word, corpus_counter, total_words):
    """
    Calculate the probability of a word given the corpus counter
    and total word count.
    """
    return corpus_counter[word] / total_words if total_words > 0 else 0

def edits1(word):
    """
    Generate all possible edits that are one edit away from the
    given word.
    """
    letters = 'abcdefghijklmnopqrstuvwxyz'
    splits = [(word[:i], word[i:]) for i in range(len(word) + 1)]
    deletes = [L + R[1:] for L, R in splits if R]
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R) > 1]
    replaces = [L + c + R[1:] for L, R in splits if R for c in letters]
    inserts = [L + c + R for L, R in splits for c in letters]
    return set(deletes + transposes + replaces + inserts)

def known(words, corpus_counter):
    """
    Return the subset of words that are actually in the corpus.
    """
    return set(w for w in words if w in corpus_counter)

def candidates(word, corpus_counter):
    """
    Generate possible correction candidates for a given word.
    """
    known_words = known([word], corpus_counter)
    known_edits1 = known(edits1(word), corpus_counter)
    known_edits2 = known([e2 for e1 in edits1(word) for e2 in edits1(e1)], corpus_counter)
    return known_words or known_edits1 or known_edits2 or [word]

def correct(word, corpus_counter, total_words):
    """
    Find the most probable correct word from the candidates.
    """
    candidates_list = candidates(word, corpus_counter)
    return max(candidates_list, key=lambda w: probability(w, corpus_counter, total_words))

def spell_checker(corpus_text, misspelled_word):
    """
    Correct the given misspelled word using the provided corpus
    text.
    """
    corpus_counter = build_corpus(corpus_text)
    total_words = sum(corpus_counter.values())
    all_edits1 = edits1(misspelled_word)
    known_edits1 = known(all_edits1, corpus_counter)
    all_edits2 = {e2 for e1 in all_edits1 for e2 in edits1(e1)}
    known_edits2 = known(all_edits2, corpus_counter)
    final_correction = correct(misspelled_word, corpus_counter, total_words)
    return final_correction, {
        'all_edits1': all_edits1,
        'known_edits1': known_edits1,
        'all_edits2': all_edits2,
        'known_edits2': known_edits2
    }

# Input from the user
corpus_text = input("Enter the corpus text: ")
misspelled_word = input("Enter the misspelled word: ")

# Correcting the misspelled word
corrected_word, edits_info = spell_checker(corpus_text, misspelled_word)

print(f"Original: {misspelled_word}")
print(f"Corrected: {corrected_word}")

# Print all possible edits
print("\nPossible Edits:")
```

```
print(f"All edits 1 away: {edits_info['all_edits1']}")
print(f"Known edits 1 away: {edits_info['known_edits1']}")
print(f"All edits 2 away: {edits_info['all_edits2']}")
print(f"Known edits 2 away: {edits_info['known_edits2']}")
```

```
Enter the corpus text: I am Vedant Deore TY Artificial Intelligence and Data Science student
Enter the misspelled word: Devant
Original: Devant
Corrected: vedant

Possible Edits:
All edits 1 away: {'eevant', 'Devadt', 'Devaknt', 'Defant', 'Dexant', 'Dfvant', 'Devanut', 'Detant', 'Dwvant', 'Devast', 'Dxevant', 'Dev
Known edits 1 away: set()
All edits 2 away: {'Dezrnt', 'Dvevankt', 'Devankta', 'Devnast', 'Dezvanz', 'Drvcant', 'zievant', 'gDevankt', 'Devadf', 'Dsevacnt', 'Dzei
Known edits 2 away: {'vedant'}
```