```
import numpy as np

# Dataset teks pertama
text1 = ["apple", "banana", "orange", "banana", "apple", "orange", "banana"]

# Dataset teks kedua
text2 = ["apple", "banana", "orange", "banana", "apple", "orange", "banana", "melon", "apple", "banana", "orange", "melon"]

# Fungsi untuk membangun kosakata dan mapping kata ke indeks dan sebaliknya
def build_vocab(text):
    vocab = sorted(set(text))
    word_to_index = {word: i for i, word in enumerate(vocab)}
    index_to_word = {i: word for i, word in enumerate(vocab)}
    return vocab, word_to_index, index_to_word

# Membangun kosakata dan mapping untuk kedua teks
vocab1, word_to_index1, index_to_word1 = build_vocab(text1)
vocab2, word_to_index2, index_to_word2 = build_vocab(text2)

# Mendapatkan ukuran kosakata
vocab_size1 = len(vocab1)
vocab_size2 = len(vocab2)

# Fungsi untuk melakukan one-hot encoding
def one_hot_encode(word, vocab_size, word_to_index):
    vector = np.zeros(vocab_size)
    vector[word_to_index[word]] = 1
    return vector

# One-hot encoding dataset
input1 = np.array([one_hot_encode(word, vocab_size1, word_to_index1) for word in text1])
input2 = np.array([one_hot_encode(word, vocab_size2, word_to_index2) for word in text2])

# Inisialisasi bobot (RNN, U, V) untuk kedua dataset
u1 = np.random.rand(vocab_size1, vocab_size1)
w1 = np.random.rand(vocab_size1, vocab_size1)
v1 = np.random.rand(vocab_size1, vocab_size1)

u2 = np.random.rand(vocab_size2, vocab_size2)
w2 = np.random.rand(vocab_size2, vocab_size2)
v2 = np.random.rand(vocab_size2, vocab_size2)

# Inisialisasi hidden state awal
h_t1 = np.zeros(vocab_size1)
h_t2 = np.zeros(vocab_size2)

# Placeholder untuk menyimpan output dan loss untuk kedua dataset
output1 = []
output2 = []
losses1 = []
losses2 = []

# Fungsi aktivasi tanh dan loss MSE
def tanh(x):
    return np.tanh(x)

def mse_loss(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

# Loop melalui setiap timestep untuk menghitung hidden state, output, dan loss untuk text1
print("Timestep | Input Word | Hidden State (h_t1) | Output (tanh) | Predicted Word | Loss")
print("--------------------------------------------------------------------------------")

for t in range(len(input1) - 1):
    x_t1 = input1[t]
    y_true1 = input1[t + 1]  # Target output adalah kata berikutnya

    # Hitung hidden state pada timestep t untuk text1
    h_t1 = tanh(np.dot(u1, x_t1) + np.dot(w1, h_t1))

    # Hitung output untuk text1
    z1 = np.dot(v1, h_t1)
    output_tanh1 = tanh(z1)

    # Hitung loss menggunakan MSE
    loss1 = mse_loss(y_true1, output_tanh1)
    losses1.append(loss1)

    # Prediksi kata dengan mencari argmax dari output_tanh1
    predicted_index1 = np.argmax(output_tanh1)
    predicted_word1 = index_to_word1[predicted_index1]
```

```
        # Simpan output
        output1.append(predicted_word1)

        # Tampilkan hasil per timestep untuk text1
        print(f"{t} | {text1[t]} | {h_t1} | {output_tanh1} | {predicted_word1} | {loss1:.4f}")

    # Loop melalui setiap timestep untuk menghitung hidden state, output, dan loss untuk text2
    print("\nTimestep | Input Word | Hidden State (h_t2) | Output (tanh) | Predicted Word | Loss")
    print("--------------------------------------------------------------------------------")

    for t in range(len(input2) - 1):
        x_t2 = input2[t]
        y_true2 = input2[t + 1]  # Target output adalah kata berikutnya

        # Hitung hidden state pada timestep t untuk text2
        h_t2 = tanh(np.dot(u2, x_t2) + np.dot(w2, h_t2))

        # Hitung output untuk text2
        z2 = np.dot(v2, h_t2)
        output_tanh2 = tanh(z2)

        # Hitung loss menggunakan MSE
        loss2 = mse_loss(y_true2, output_tanh2)
        losses2.append(loss2)

        # Prediksi kata dengan mencari argmax dari output_tanh2
        predicted_index2 = np.argmax(output_tanh2)
        predicted_word2 = index_to_word2[predicted_index2]

        # Simpan output
        output2.append(predicted_word2)

        # Tampilkan hasil per timestep untuk text2
        print(f"{t} | {text2[t]} | {h_t2} | {output_tanh2} | {predicted_word2} | {loss2:.4f}")

    # Menampilkan hasil prediksi kata-kata untuk text1 dan text2
    print("\nHasil prediksi kata-kata untuk text1:")
    print(output1)

    print("\nHasil prediksi kata-kata untuk text2:")
    print(output2)

    # Menampilkan nilai rata-rata loss untuk text1 dan text2
    average_loss1 = np.mean(losses1)
    average_loss2 = np.mean(losses2)
    print(f"\nRata-rata Loss untuk text1: {average_loss1:.4f}")
    print(f"Rata-rata Loss untuk text2: {average_loss2:.4f}")
```

```
Timestep | Input Word | Hidden State (h_t1) | Output (tanh) | Predicted Word | Loss
--------------------------------------------------------------------------------
0 | apple | [0.61267491 0.23681643 0.24447585] | [0.34579573 0.39106116 0.5211239 ] | orange | 0.2540
1 | banana | [0.68864419 0.81261772 0.85635829] | [0.70519231 0.62315567 0.87197713] | orange | 0.3007
2 | orange | [0.94147618 0.93648875 0.95243002] | [0.77599137 0.71837508 0.91911033] | orange | 0.5087
3 | banana | [0.94948534 0.95219694 0.97752983] | [0.78351269 0.72502823 0.92381784] | orange | 0.4753
4 | apple | [0.98727459 0.89992458 0.96072324] | [0.77607678 0.72683604 0.92182697] | orange | 0.3789
5 | orange | [0.97193698 0.95876618 0.97495652] | [0.78638296 0.73045502 0.92528316] | orange | 0.5157

Timestep | Input Word | Hidden State (h_t2) | Output (tanh) | Predicted Word | Loss
--------------------------------------------------------------------------------
0 | apple | [0.62183682 0.03482126 0.23068847 0.54884646] | [0.75996812 0.59510882 0.62632447 0.4710115 ] | apple | 0.3389
1 | banana | [0.80506238 0.86011926 0.66129907 0.91542488] | [0.96860612 0.87026898 0.92681078 0.92808621] | apple | 0.6399
2 | orange | [0.85690155 0.89019973 0.94249417 0.98319955] | [0.9817587  0.89086635 0.94106744 0.96132112] | apple | 0.6964
3 | banana | [0.92141638 0.95582789 0.91347692 0.9950243 ] | [0.98370224 0.90350615 0.95258421 0.96473018] | apple | 0.6637
4 | apple | [0.93818452 0.92216767 0.86233785 0.99517222] | [0.98245212 0.90242099 0.95123916 0.95962351] | apple | 0.6715
5 | orange | [0.89021726 0.92282627 0.95640306 0.99072832] | [0.98340227 0.8978916  0.94754615 0.96503512] | apple | 0.7017
6 | banana | [0.92558554 0.95922231 0.91745001 0.99562909] | [0.98391699 0.90424312 0.95324248 0.96528458] | apple | 0.6799
7 | melon | [0.77367016 0.94441113 0.96809307 0.99777654] | [0.9821212  0.88798061 0.93681167 0.96389249] | apple | 0.6489
8 | apple | [0.92886156 0.8985896  0.86245814 0.9940753 ] | [0.98187024 0.89969613 0.94858712 0.95788947] | apple | 0.6979
9 | banana | [0.9250101  0.9616508  0.9101433  0.99528945] | [0.9837903  0.9042598  0.95328572 0.96490898] | apple | 0.6739
10 | orange | [0.89234799 0.92209756 0.95925161 0.99156556] | [0.98351052 0.89817902 0.94774475 0.96524468] | apple | 0.6771

Hasil prediksi kata-kata untuk text1:
['orange', 'orange', 'orange', 'orange', 'orange', 'orange']

Hasil prediksi kata-kata untuk text2:
['apple', 'apple', 'apple', 'apple', 'apple', 'apple', 'apple', 'apple', 'apple', 'apple', 'apple']

Rata-rata Loss untuk text1: 0.4056
Rata-rata Loss untuk text2: 0.6445
```