https://github.com/wazowz/asss-last

```
import numpy as np

sentences = [
    ["i", "like", "deep", "learning"],
    ["i", "like", "nlp", "projects"],
    ["i", "enjoy", "deep", "projects"]
]

vocab = list(set(word for sent in sentences for word in sent))
word_to_index = {w: i for i, w in enumerate(vocab)}
index_to_word = {i: w for w, i in word_to_index.items()}

vocab_size = len(vocab)
hidden_size = 10
learning_rate = 0.01

Wxh = np.random.randn(hidden_size, vocab_size) * 0.01
Whh = np.random.randn(hidden_size, hidden_size) * 0.01
Why = np.random.randn(vocab_size, hidden_size) * 0.01
bh = np.zeros((hidden_size, 1))
by = np.zeros((vocab_size, 1))

def softmax(x):
    e_x = np.exp(x - np.max(x))
    return e_x / np.sum(e_x)

def word_to_onehot(word):
    vec = np.zeros((vocab_size, 1))
    vec[word_to_index[word]] = 1
    return vec

for epoch in range(5000):
    total_loss = 0
    for sentence in sentences:
        inputs = [word_to_onehot(w) for w in sentence[:-1]]
        target = word_to_index[sentence[-1]]

        hs = {}
        hs[-1] = np.zeros((hidden_size, 1))
```

```
    for t in range(len(inputs)):
        hs[t] = np.tanh(np.dot(Wxh, inputs[t]) + np.dot(Whh, hs[t-1]) + bh)

    y = np.dot(Why, hs[2]) + by
    p = softmax(y)


    loss = -np.log(p[target][0])
    total_loss += loss


    dy = p
    dy[target] -= 1

    dWhy = np.dot(dy, hs[2].T)
    dby = dy

    dh = np.dot(Why.T, dy)
    dWxh = np.zeros_like(Wxh)
    dWhh = np.zeros_like(Whh)
    dbh = np.zeros_like(bh)

    for t in reversed(range(len(inputs))):
        dtanh = (1 - hs[t] ** 2) * dh
        dWxh += np.dot(dtanh, inputs[t].T)
        dWhh += np.dot(dtanh, hs[t-1].T)
        dbh += dtanh
        dh = np.dot(Whh.T, dtanh)


    for dparam in [dWxh, dWhh, dWhy, dbh, dby]:
        np.clip(dparam, -5, 5, out=dparam)

    Wxh -= learning_rate * dWxh
    Whh -= learning_rate * dWhh
    Why -= learning_rate * dWhy
    bh -= learning_rate * dbh
    by -= learning_rate * dby
```

```
    if epoch % 500 == 0:
        print(f"Epoch {epoch}, Loss: {total_loss:.4f}")


def predict(word1, word2, word3):
    x1, x2, x3 = map(word_to_onehot, [word1, word2, word3])
    h1 = np.tanh(np.dot(Wxh, x1) + bh)
    h2 = np.tanh(np.dot(Wxh, x2) + np.dot(Whh, h1) + bh)
    h3 = np.tanh(np.dot(Wxh, x3) + np.dot(Whh, h2) + bh)
    y = np.dot(Why, h3) + by
    p = softmax(y)
    idx = np.argmax(p)
    return index_to_word[idx]


print("Prediction for ['i', 'like', 'deep']:", predict("i", "like", "deep"))
print("Prediction for ['i', 'enjoy', 'deep']:", predict("i", "enjoy", "deep"))
```