NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Assignment for SC4002 / CE4045 / CZ4045

AY2023-2024

Group ID: G45

Group members:

Name	Matric No.
Kng Yew Chian	U2021777G
Raghav Rajendran Nair	U2023420G
Peh Wei Hang	U2022903K
Koh Jin Kiong Brian	U2022245E

Part 1. Sequence Tagging: NER

Question 1.1

Based on word2vec embeddings you have downloaded, use cosine similarity to find the most similar word to each of these words: (a) "student"; (b) "Apple"; (c) "apple". Report the most similar word and its cosine similarity.

a) "student"

Most similar word	Cosine similarity
students	0.7294867038726807

b) "Apple"

Most similar word	Cosine similarity
Apple_AAPL	0.7456986308097839

c) "apple"

Most similar word	Cosine similarity
apples	0.720359742641449

Code Snippet:

```
In [5]: words = ["student", "Apple", "apple"]
        print("-----------------------------------------------------------------------------")
        print("Word\t\tMost similar word\tCosine similarity")
        print("-----------------------------------------------------------------------------")
        for word in words:
            most_similar = embeddings.most_similar(positive=[word])
            print(f"{word}\t\t{most_similar[0][0]}  \t\t{most_similar[0][1]}")
        print("-----------------------------------------------------------------------------")
```

```
-----------------------------------------------------------------------
Word            Most similar word       Cosine similarity
-----------------------------------------------------------------------
student         students                0.7294867038726807
Apple           Apple_AAPL              0.7456986308097839
apple           apples                  0.720359742641449
-----------------------------------------------------------------------
```

Question 1.2(a)

Describe the size (number of sentences) of the training, development and test file for CoNLL2003. Specify the complete set of all possible word labels based on the tagging scheme (IO, BIO,etc.) you chose.

Size of training, development and test file:

Size of training file:	14987 sentences
Size of development file:	3466 sentences
Size of test file:	3684 sentences

Code Snippet:

```
In [11]: print("Number of sentences (training):", len(train_sentences))
         print("Number of sentences (dev):", len(dev_sentences))
         print("Number of sentences (test):", len(test_sentences))

         Number of sentences (training): 14987
         Number of sentences (dev): 3466
         Number of sentences (test): 3684
```

Complete set of all possible word labels

The dataset follows the BIO (IOB1) tagging scheme, which we did not alter.

Tag set (BIO): ['B-LOC' 'B-MISC' 'B-ORG' 'I-LOC' 'I-MISC' 'I-ORG' 'I-PER' 'O']

Code Snippet:

```
In [12]: print("Tag set (BIO):", tag_set)

         Tag set (BIO): ['B-LOC' 'B-MISC' 'B-ORG' 'I-LOC' 'I-MISC' 'I-ORG' 'I-PER' 'O']
```

Question 1.2(b)

Choose an example sentence from the training set of CoNLL2003 that has at least two named entities with more than one word. Explain how to form complete named entities from the label for each word, and list all the named entities in this sentence.

Example sentence:

"Swiss Grand Prix World Cup cycling race on Sunday:"

List of all named entities in example sentence:

['Swiss', 'Grand Prix', 'World Cup']

Explanation:

Words tagged with "O" are outside of named entities and the "I-XXX" tag is used for words inside a named entity of type XXX. Whenever two entities of type XXX are immediately next to each other, the first word of the second entity will be tagged "B-XXX" in order to show that it starts another entity. Thus, we can form complete named entities (more than one word) by getting a word with tag "B-XXX" (e.g B-MISC) and combining it with the subsequent "I-XXX" (e.g I-MISC) tagged words until a different tag is reached. This is shown in the code snippet attached below.

In the case of the above example sentence, there are two named entities with more than one word. They are "Grand Prix" and "World Cup".

Code Snippet:

In [14]:
```
sentence = get_multiple_ne_sentence(train_sentences)
sentence
```

Out[14]:
```
[['Swiss', 'I-MISC'],
 ['Grand', 'B-MISC'],
 ['Prix', 'I-MISC'],
 ['World', 'B-MISC'],
 ['Cup', 'I-MISC'],
 ['cycling', 'O'],
 ['race', 'O'],
 ['on', 'O'],
 ['Sunday', 'O'],
 [':', 'O']]
```

In [15]:
```
def get_named_entities(sentence):
    inside_tags = ['I-ORG', 'I-LOC', 'I-PER', 'I-MISC'] # Tags that require multiple words to form an entity
    begin_tags = ['B-LOC', 'B-ORG', 'B-MISC'] # Tags that are single word entities
    outside_tags = ['O']
    entities = [] # all entities gotten from search
    entity = [] # word group of current entity if any group tags encountered

    for c in sentence:
        if (c['tag'] in begin_tags or c['tag'] in outside_tags or c['tag'] == '\n') and len(entity) != 0:
            entities.append(' '.join(entity))
            entity = []
        if c['tag'] in begin_tags or c['tag'] in inside_tags:
            entity.append(c['text'])

    return entities
```

In [16]:
```
_,_,sentence_text_tag = split_text_tag([sentence])
print("Complete named entities in the sentence:", get_named_entities(sentence_text_tag))
```

```
Complete named entities in the sentence: ['Swiss', 'Grand Prix', 'World Cup']
```

Question 1.3(a)

Discuss how you deal with new words in the training set which are not found in the pretrained dictionary. Likewise, how do you deal with new words in the test set which are not found in either the pretrained dictionary or the training set? Show the corresponding code snippet.

We create an average embedding vector to replace Out-Of-Vocabulary words. It is the average of all the embeddings in the word2vec vocabulary provided. This provides all Out-Of-Vocabulary words with a neutral meaning.

```
# create average embedding to replace Out-Of-Vocabulary words
average_embedding = np.mean(embeddings.vectors, axis=0)

if word not in embeddings:
  new_embedding = average_embedding
```

Code Snippet:

```
In [42]: # create average embedding to replace Out-Of-Vocabulary words
         average_embedding = np.mean(embeddings.vectors, axis=0)
```

```
# replace words with embeddings and tags with integers
embeddings_in_sentences = []
new_sentence = []

count = 0
for sentence in sentences_wordstags_array:
  for word, tag in sentence:
    if tag not in tag_to_integer_dictionary:
      #print(f"{tag} not found with {word}, skipping")
      continue
    if word not in embeddings:
      #new_embedding = np.zeros(300)
      new_embedding = average_embedding

    else:
      unnormalized_embedding = embeddings[word].astype(np.float32)
      # Reshape the embedding to be a 2D array with a single row
      embedding_reshaped = unnormalized_embedding.reshape(1, -1)
      # Normalize the embedding
      embedding_normalized = normalize(embedding_reshaped, axis=1, norm='l2')
      # Flatten the normalized embedding back into a 1D array
      new_embedding = embedding_normalized.flatten()

    new_sentence.append([new_embedding, tag_to_integer_dictionary[tag]])
    count += 1

  embeddings_in_sentences.append(new_sentence)
  new_sentence = []

return embeddings_in_sentences
```

Question 1.3(b)

Describe what neural network you used to produce the final vector representation of each word and what are the mathematical functions used for the forward computation (i.e., from the pre trained word vectors to the final label of each word). Give the detailed setting of the network including which parameters are being updated, what are their sizes, and what is the length of the final vector representation of each word to be fed to the softmax classifier.

The neural network transforms pre-trained word vectors into categorical labels through a bidirectional LSTM and dense layers.

Components:
Input Layer: Receives input of shape (None, maximum sentence length(words), word2vec embedding dimensions). Note the maximum sentence length is also the timesteps in this context.

Masking Layer:
Applies a mask to the input where any timestep with a value of zero is ignored, preventing padding from affecting the subsequent layers' computations.

Bidirectional LSTM Layer:
This layer utilizes two LSTM layers that process the data in both forward and reverse directions. Each LSTM layer comprises 16 units and implements the following mathematical operations:

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Cell State:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Hidden State:

$$ht = o_t \cdot \tanh(c_t)$$

Where σ denotes the sigmoid function, tanh is the hyperbolic tangent activation function, W and b are the weights and biases of the respective gates, and * denotes element-wise multiplication.

The bidirectional wrapper concatenates the outputs from both directions for each timestep, resulting in a 32-dimensional vector.

Dense Output Layer: Applies a linear transformation followed by a softmax activation to the LSTM outputs to obtain the probability distribution over classes:

Linear Transformation:

$$z = W_d \cdot h + b_d$$

Here, W_d and b_d are the weights and biases of the Dense layer, and h is the output from the Bidirectional LSTM.

Softmax Activation:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^{\text{num\_classes}} \exp(z_j)} \quad \text{for } i = 1 \text{ to num\_classes}$$

Training Details:
Loss Function: Utilizes a custom masked_loss_function which computes the cross-entropy loss for unmasked timesteps while excluding the effects of timesteps with a label of 999.

Optimizer:
Adam optimizer with a learning rate of 0.001.

Training Process:
The model is compiled and trained over a specified number of epochs with batch-based updates, where parameters are adjusted to minimize the custom loss function, with accuracy serving as the performance metric.

Final Vector Representation:
Before classification, each word is represented by a vector of length equal to num_classes, which is the output of the dense layer. This vector encodes the probability of each class given the context of the word as understood by the bidirectional LSTM.

Length of final vector representation of each sentence: 2*Hidden Size = 2*16 = 32

Size of parameters updated:
Bi-directional LSTM

Parameters	**Size**
For one layer	
W (2 * hidden_size * embedding_dim)	2*16*300 = 9600
U (2 * hidden_size * hidden_size)	2*16*16 = 512
Biases (2 * hidden_size)	2*8 = 16

Softmax Classifier (Linear Layer)

Parameters	**Size**
Layer 1 weights ((2* LSTM hidden_size) * LSTM hidden_size)	2 * 16 * 8 = 256
Layer 1 biases (LSTM hidden_size)	8

Question 1.3(c)

Report how many epochs you used for training, as well as the running time.

Epochs	Running Time (s)
33	985

Question 1.3(d)

Report the f1 score on the test set, as well as the f1 score on the development set for each epoch during training.

Test Set F1 Score:

```
F1 Score: 0.8103161476349263
```

Train and Development Set F1 Scores:

Epoch	Train F1 Score	Development F1 Score
1	0.4958	0.5030
2	0.7178	0.7221
3	0.7642	0.7696

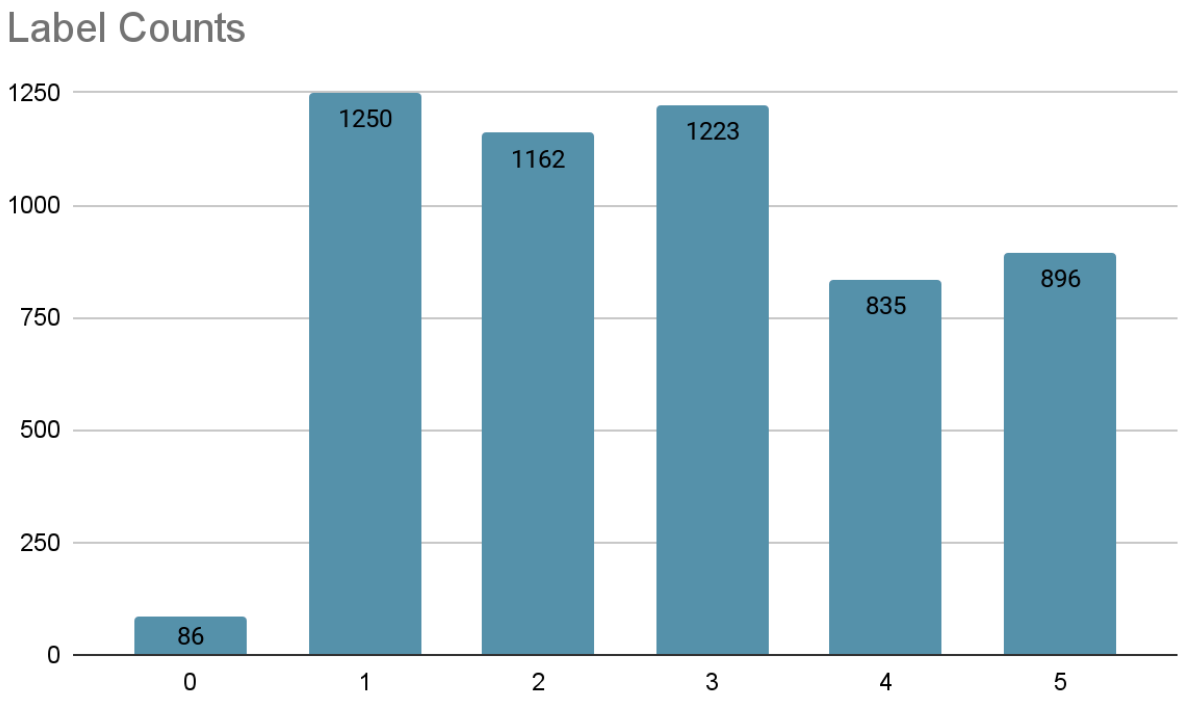
4	0.7903	07934
5	0.8116	0.8102
6	0.8226	0.8145
7	0.8340	0.8269
8	0.8399	0.8313
9	0.8492	0.8395
10	0.8574	0.8440
11	0.8610	0.8418
12	0.8667	0.8497
13	0.8723	0.8508
14	0.8783	0.8557
15	0.8744	0.8534
16	0.8877	0.8595
17	0.8906	0.8623
18	0.8943	0.8633
19	0.8915	0.8565
20	0.9014	0.8643
21	0.9069	0.8639
22	0.9038	0.8638
23	0.9107	0.8638
24	0.9090	0.8668
25	0.9179	0.8663
26	0.9182	0.8677
27	0.9217	0.8684
28	0.9208	0.8708
29	0.9269	0.8664
30	0.9301	0.8677

31	0.9333	0.8687
32	0.9326	0.8702
33	0.9380	0.8685

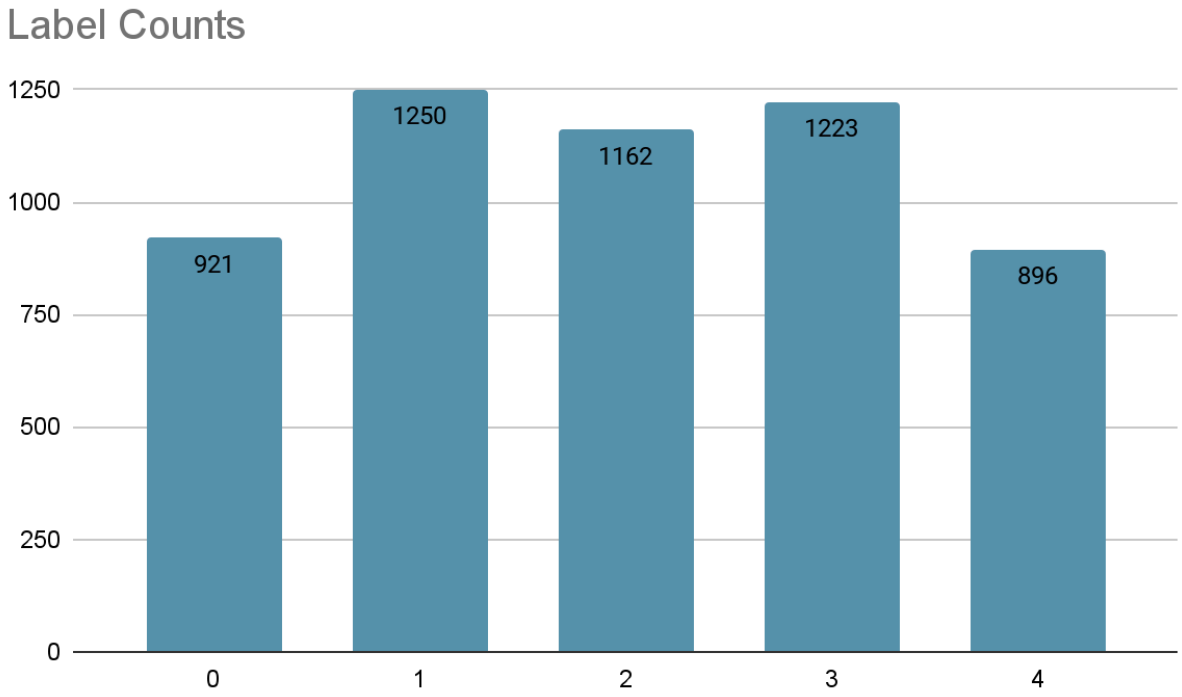
Part 2. Sentence-Level Categorization: Question Classification

Question 2(a)

Initial class label counts:



We have decided to combine classes ‘0’ and ‘4’ to form a single class ‘0’ (OTHERS) as well as renaming the class ‘5’ to class ‘4’. This helps us achieve a more uniformly distributed dataset. The resulting class labels and counts are shown below



The training data is then split into train and development sets. The development set has 500 samples with labels from the same distribution as the train set.

Question 2(b)

Single direction and bi-directional LSTMs were tried as aggregation methods. Both models were tuned with respect to the development set. The hyper-parameters tuned for 10 trials using the TPE (Tree-structured Parzen Estimator) algorithm. The hyper-parameters tuned were:

- learning rate – logarithmic steps between 0.1 and 0.00001
- batch size – 2, 4, 8, 16, 32, 64, 128
- number of LSTM layers – uniform distribution between 1 and 20
- hidden size of LSTM cells – uniform distribution between 1 and 20
- dropout of the LSTM cells – uniform distribution between 0 and 0.3

Each run was run to the maximum of 500 epochs. Early stopping was implemented with a patience of 100. The best model at the epoch with the best development accuracy was saved.

Comparing the best models of both methods after hyperparameter tuning, bi-directional LSTM has better accuracy of **92.8%** compared to the single direction LSTM with accuracy of **90.2%** As a result, bi-directional LSTM is chosen as the aggregate function.

Question 2(c)

The neural network we have chosen converts word embeddings of inputs to its final representation (categorical labels) by feeding the inputs through a bi-directional lstm layer followed by 2 linear and 1 non-linear layer.

The Bi-Directional LSTM Layer: Bi-Directional LSTM makes use of 2 LSTM layers to feed the data both forwards and backwards. Firstly, the pre-trained word vectors (x_t) are fed into the forget gate(f_t), the current cell state(C_t), the input gate(i_t) and the output gate(o_t). Next, the forget gate(f_t) multiplies with the previous cell state(C_{t-1}) to erase some information but also brings in new information by multiplying the input gate(i_t) with the current cell state(C_t) to update the current cell state(C_t). Finally, the tanh activation function is applied on the updated cell state(C_t) which is then multiplied with the output gate(o_t) to get the hidden state output. A similar process happens for the reverse direction except the hidden states and cell states received are from timestamp t + 1 instead of t - 1.

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Cell State:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Hidden State:

$$ht = o_t \cdot \tanh(c_t)$$

Softmax Classifier: the output is then passed through a linear transformation(z) followed by a non-linear Relu activation function(f(x)) followed by another linear transformation(z) and finally a softmax activation function to finally produce a probability distribution over the different labels.

Linear Transformation:

$$z = W_d \cdot h + b_d$$

ReLU activation function:

$$f(x) = max(0, x)$$

Softmax Classifier:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^{\text{num\_classes}} \exp(z_j)} \quad \text{for } i = 1 \text{ to num\_classes}$$

During the training of bi-diretional LSTM, the weight(W) and biases(b) of the respective gates are being updated through backpropagation and gradient descent optimisation methods with the goal of minimizing the loss function.

Loss function used: Cross Entropy Loss.
Optimiser used: Adam
Length of final vector representation of each sentence: 2*Hidden Size = 2*8 = 16

Size of parameters updated:

Bi-directional LSTM

Parameters	**Size**
For one layer	
W (2 * hidden_size * embedding_dim)	2*8*300 = 4800
U (2 * hidden_size * hidden_size)	2*8*8 = 128
Biases (2 * hidden_size)	2*8 = 16
Total for all 4 layers	
W	19200
U	512
biases	64

Softmax Classifier (Linear Layer)

Parameters	**Size**
Layer 1 weights ((2* LSTM hidden_size) * LSTM hidden_size)	2 * 8 * 8 = 128
Layer 1 biases (LSTM hidden_size)	8
Layer 2 weights	8 * 6 = 48
Layer 2 biases	6

Question 2(d)

The best model ran for 240 epochs over 19 minutes and 31 seconds.

Hyperparameters:

Parameter	**Value**
Learning rate	0.005151046611453035
Layers	4
Hidden Size	8
Dropout	0.18

Question 2(e)

Accuracy on test set: **90.0%**

Accuracy on development set for each epoch

[2023-11-10 14:03:14,023][__main__][INFO] - Epoch 1 dev_accuracy: 51.80000
[2023-11-10 14:03:21,042][__main__][INFO] - Epoch 2 dev_accuracy: 63.00000
[2023-11-10 14:03:28,221][__main__][INFO] - Epoch 3 dev_accuracy: 70.20000
[2023-11-10 14:03:35,234][__main__][INFO] - Epoch 4 dev_accuracy: 73.40000
[2023-11-10 14:03:42,248][__main__][INFO] - Epoch 5 dev_accuracy: 81.80000
[2023-11-10 14:03:49,306][__main__][INFO] - Epoch 6 dev_accuracy: 84.20000
[2023-11-10 14:03:56,338][__main__][INFO] - Epoch 7 dev_accuracy: 85.40000
[2023-11-10 14:04:03,420][__main__][INFO] - Epoch 8 dev_accuracy: 85.20000
[2023-11-10 14:04:08,097][__main__][INFO] - Epoch 9 dev_accuracy: 85.60000
[2023-11-10 14:04:15,209][__main__][INFO] - Epoch 10 dev_accuracy: 88.80000
[2023-11-10 14:04:22,466][__main__][INFO] - Epoch 11 dev_accuracy: 88.40000
[2023-11-10 14:04:27,215][__main__][INFO] - Epoch 12 dev_accuracy: 89.20000
[2023-11-10 14:04:34,229][__main__][INFO] - Epoch 13 dev_accuracy: 89.00000
[2023-11-10 14:04:38,888][__main__][INFO] - Epoch 14 dev_accuracy: 89.60000
[2023-11-10 14:04:45,949][__main__][INFO] - Epoch 15 dev_accuracy: 89.00000
[2023-11-10 14:04:50,604][__main__][INFO] - Epoch 16 dev_accuracy: 89.80000
[2023-11-10 14:04:57,481][__main__][INFO] - Epoch 17 dev_accuracy: 90.00000
[2023-11-10 14:05:04,478][__main__][INFO] - Epoch 18 dev_accuracy: 90.80000
[2023-11-10 14:05:11,443][__main__][INFO] - Epoch 19 dev_accuracy: 88.80000
[2023-11-10 14:05:16,047][__main__][INFO] - Epoch 20 dev_accuracy: 91.20000
[2023-11-10 14:05:23,169][__main__][INFO] - Epoch 21 dev_accuracy: 91.80000
[2023-11-10 14:05:30,231][__main__][INFO] - Epoch 22 dev_accuracy: 89.80000
[2023-11-10 14:05:34,804][__main__][INFO] - Epoch 23 dev_accuracy: 88.20000
[2023-11-10 14:05:39,341][__main__][INFO] - Epoch 24 dev_accuracy: 90.60000
[2023-11-10 14:05:43,912][__main__][INFO] - Epoch 25 dev_accuracy: 89.80000

[2023-11-10 14:05:48,544][__main__][INFO] - Epoch 26 dev_accuracy: 91.00000
[2023-11-10 14:05:53,140][__main__][INFO] - Epoch 27 dev_accuracy: 91.40000
[2023-11-10 14:05:57,704][__main__][INFO] - Epoch 28 dev_accuracy: 91.40000
[2023-11-10 14:06:02,370][__main__][INFO] - Epoch 29 dev_accuracy: 90.40000
[2023-11-10 14:06:06,872][__main__][INFO] - Epoch 30 dev_accuracy: 90.00000
[2023-11-10 14:06:11,542][__main__][INFO] - Epoch 31 dev_accuracy: 91.60000
[2023-11-10 14:06:16,049][__main__][INFO] - Epoch 32 dev_accuracy: 91.20000
[2023-11-10 14:06:20,616][__main__][INFO] - Epoch 33 dev_accuracy: 91.20000
[2023-11-10 14:06:25,191][__main__][INFO] - Epoch 34 dev_accuracy: 90.20000
[2023-11-10 14:06:29,835][__main__][INFO] - Epoch 35 dev_accuracy: 90.60000
[2023-11-10 14:06:34,467][__main__][INFO] - Epoch 36 dev_accuracy: 90.60000
[2023-11-10 14:06:39,041][__main__][INFO] - Epoch 37 dev_accuracy: 91.00000
[2023-11-10 14:06:43,651][__main__][INFO] - Epoch 38 dev_accuracy: 89.00000
[2023-11-10 14:06:48,276][__main__][INFO] - Epoch 39 dev_accuracy: 89.80000
[2023-11-10 14:06:52,892][__main__][INFO] - Epoch 40 dev_accuracy: 89.40000
[2023-11-10 14:06:57,489][__main__][INFO] - Epoch 41 dev_accuracy: 90.20000
[2023-11-10 14:07:02,008][__main__][INFO] - Epoch 42 dev_accuracy: 90.20000
[2023-11-10 14:07:06,593][__main__][INFO] - Epoch 43 dev_accuracy: 88.60000
[2023-11-10 14:07:11,365][__main__][INFO] - Epoch 44 dev_accuracy: 88.60000
[2023-11-10 14:07:16,105][__main__][INFO] - Epoch 45 dev_accuracy: 91.00000
[2023-11-10 14:07:20,867][__main__][INFO] - Epoch 46 dev_accuracy: 89.20000
[2023-11-10 14:07:25,593][__main__][INFO] - Epoch 47 dev_accuracy: 90.00000
[2023-11-10 14:07:30,357][__main__][INFO] - Epoch 48 dev_accuracy: 90.40000
[2023-11-10 14:07:34,971][__main__][INFO] - Epoch 49 dev_accuracy: 90.80000
[2023-11-10 14:07:39,762][__main__][INFO] - Epoch 50 dev_accuracy: 89.20000
[2023-11-10 14:07:44,416][__main__][INFO] - Epoch 51 dev_accuracy: 90.40000
[2023-11-10 14:07:49,076][__main__][INFO] - Epoch 52 dev_accuracy: 89.80000
[2023-11-10 14:07:53,695][__main__][INFO] - Epoch 53 dev_accuracy: 90.40000
[2023-11-10 14:07:58,312][__main__][INFO] - Epoch 54 dev_accuracy: 90.40000
[2023-11-10 14:08:02,919][__main__][INFO] - Epoch 55 dev_accuracy: 91.00000
[2023-11-10 14:08:07,657][__main__][INFO] - Epoch 56 dev_accuracy: 90.60000
[2023-11-10 14:08:12,301][__main__][INFO] - Epoch 57 dev_accuracy: 89.20000
[2023-11-10 14:08:17,029][__main__][INFO] - Epoch 58 dev_accuracy: 90.40000
[2023-11-10 14:08:21,766][__main__][INFO] - Epoch 59 dev_accuracy: 91.40000
[2023-11-10 14:08:26,465][__main__][INFO] - Epoch 60 dev_accuracy: 90.20000
[2023-11-10 14:08:31,261][__main__][INFO] - Epoch 61 dev_accuracy: 91.00000
[2023-11-10 14:08:36,014][__main__][INFO] - Epoch 62 dev_accuracy: 91.60000
[2023-11-10 14:08:40,780][__main__][INFO] - Epoch 63 dev_accuracy: 91.60000
[2023-11-10 14:08:45,577][__main__][INFO] - Epoch 64 dev_accuracy: 92.00000
[2023-11-10 14:08:52,593][__main__][INFO] - Epoch 65 dev_accuracy: 91.80000
[2023-11-10 14:08:57,222][__main__][INFO] - Epoch 66 dev_accuracy: 90.80000
[2023-11-10 14:09:01,828][__main__][INFO] - Epoch 67 dev_accuracy: 88.00000
[2023-11-10 14:09:06,370][__main__][INFO] - Epoch 68 dev_accuracy: 91.80000
[2023-11-10 14:09:11,033][__main__][INFO] - Epoch 69 dev_accuracy: 91.40000
[2023-11-10 14:09:15,692][__main__][INFO] - Epoch 70 dev_accuracy: 91.20000
[2023-11-10 14:09:20,354][__main__][INFO] - Epoch 71 dev_accuracy: 91.20000
[2023-11-10 14:09:24,915][__main__][INFO] - Epoch 72 dev_accuracy: 91.80000
[2023-11-10 14:09:29,588][__main__][INFO] - Epoch 73 dev_accuracy: 89.80000
[2023-11-10 14:09:34,255][__main__][INFO] - Epoch 74 dev_accuracy: 90.00000
[2023-11-10 14:09:38,903][__main__][INFO] - Epoch 75 dev_accuracy: 91.40000
[2023-11-10 14:09:43,589][__main__][INFO] - Epoch 76 dev_accuracy: 90.80000
[2023-11-10 14:09:48,302][__main__][INFO] - Epoch 77 dev_accuracy: 90.40000
[2023-11-10 14:09:53,000][__main__][INFO] - Epoch 78 dev_accuracy: 91.20000
[2023-11-10 14:09:57,698][__main__][INFO] - Epoch 79 dev_accuracy: 91.80000

[2023-11-10 14:10:02,301][__main__][INFO] - Epoch 80 dev_accuracy: 90.80000
[2023-11-10 14:10:07,039][__main__][INFO] - Epoch 81 dev_accuracy: 91.40000
[2023-11-10 14:10:11,699][__main__][INFO] - Epoch 82 dev_accuracy: 91.00000
[2023-11-10 14:10:16,430][__main__][INFO] - Epoch 83 dev_accuracy: 90.80000
[2023-11-10 14:10:21,103][__main__][INFO] - Epoch 84 dev_accuracy: 91.20000
[2023-11-10 14:10:25,618][__main__][INFO] - Epoch 85 dev_accuracy: 90.00000
[2023-11-10 14:10:30,340][__main__][INFO] - Epoch 86 dev_accuracy: 91.00000
[2023-11-10 14:10:34,987][__main__][INFO] - Epoch 87 dev_accuracy: 92.20000
[2023-11-10 14:10:42,024][__main__][INFO] - Epoch 88 dev_accuracy: 91.40000
[2023-11-10 14:10:46,488][__main__][INFO] - Epoch 89 dev_accuracy: 91.60000
[2023-11-10 14:10:51,113][__main__][INFO] - Epoch 90 dev_accuracy: 91.60000
[2023-11-10 14:10:55,642][__main__][INFO] - Epoch 91 dev_accuracy: 92.00000
[2023-11-10 14:11:00,150][__main__][INFO] - Epoch 92 dev_accuracy: 91.80000
[2023-11-10 14:11:04,941][__main__][INFO] - Epoch 93 dev_accuracy: 92.00000
[2023-11-10 14:11:09,460][__main__][INFO] - Epoch 94 dev_accuracy: 91.40000
[2023-11-10 14:11:14,075][__main__][INFO] - Epoch 95 dev_accuracy: 89.80000
[2023-11-10 14:11:18,597][__main__][INFO] - Epoch 96 dev_accuracy: 91.00000
[2023-11-10 14:11:23,218][__main__][INFO] - Epoch 97 dev_accuracy: 91.60000
[2023-11-10 14:11:27,986][__main__][INFO] - Epoch 98 dev_accuracy: 91.20000
[2023-11-10 14:11:32,566][__main__][INFO] - Epoch 99 dev_accuracy: 91.40000
[2023-11-10 14:11:37,187][__main__][INFO] - Epoch 100 dev_accuracy: 90.60000
[2023-11-10 14:11:41,818][__main__][INFO] - Epoch 101 dev_accuracy: 90.20000
[2023-11-10 14:11:46,398][__main__][INFO] - Epoch 102 dev_accuracy: 90.80000
[2023-11-10 14:11:50,962][__main__][INFO] - Epoch 103 dev_accuracy: 90.00000
[2023-11-10 14:11:55,540][__main__][INFO] - Epoch 104 dev_accuracy: 90.40000
[2023-11-10 14:12:00,178][__main__][INFO] - Epoch 105 dev_accuracy: 90.40000
[2023-11-10 14:12:04,878][__main__][INFO] - Epoch 106 dev_accuracy: 91.00000
[2023-11-10 14:12:09,558][__main__][INFO] - Epoch 107 dev_accuracy: 90.40000
[2023-11-10 14:12:14,161][__main__][INFO] - Epoch 108 dev_accuracy: 90.40000
[2023-11-10 14:12:18,721][__main__][INFO] - Epoch 109 dev_accuracy: 91.40000
[2023-11-10 14:12:23,310][__main__][INFO] - Epoch 110 dev_accuracy: 90.00000
[2023-11-10 14:12:27,957][__main__][INFO] - Epoch 111 dev_accuracy: 90.60000
[2023-11-10 14:12:32,575][__main__][INFO] - Epoch 112 dev_accuracy: 91.00000
[2023-11-10 14:12:37,250][__main__][INFO] - Epoch 113 dev_accuracy: 88.40000
[2023-11-10 14:12:42,038][__main__][INFO] - Epoch 114 dev_accuracy: 88.60000
[2023-11-10 14:12:46,723][__main__][INFO] - Epoch 115 dev_accuracy: 91.40000
[2023-11-10 14:12:51,436][__main__][INFO] - Epoch 116 dev_accuracy: 90.80000
[2023-11-10 14:12:56,041][__main__][INFO] - Epoch 117 dev_accuracy: 89.00000
[2023-11-10 14:13:00,694][__main__][INFO] - Epoch 118 dev_accuracy: 91.20000
[2023-11-10 14:13:05,350][__main__][INFO] - Epoch 119 dev_accuracy: 90.80000
[2023-11-10 14:13:10,160][__main__][INFO] - Epoch 120 dev_accuracy: 90.20000
[2023-11-10 14:13:14,746][__main__][INFO] - Epoch 121 dev_accuracy: 90.80000
[2023-11-10 14:13:19,258][__main__][INFO] - Epoch 122 dev_accuracy: 90.40000
[2023-11-10 14:13:23,930][__main__][INFO] - Epoch 123 dev_accuracy: 90.20000
[2023-11-10 14:13:28,531][__main__][INFO] - Epoch 124 dev_accuracy: 91.20000
[2023-11-10 14:13:33,219][__main__][INFO] - Epoch 125 dev_accuracy: 91.80000
[2023-11-10 14:13:37,895][__main__][INFO] - Epoch 126 dev_accuracy: 92.00000
[2023-11-10 14:13:42,639][__main__][INFO] - Epoch 127 dev_accuracy: 91.20000
[2023-11-10 14:13:47,293][__main__][INFO] - Epoch 128 dev_accuracy: 91.20000
[2023-11-10 14:13:51,790][__main__][INFO] - Epoch 129 dev_accuracy: 91.60000
[2023-11-10 14:13:56,448][__main__][INFO] - Epoch 130 dev_accuracy: 91.00000
[2023-11-10 14:14:01,088][__main__][INFO] - Epoch 131 dev_accuracy: 91.40000
[2023-11-10 14:14:05,662][__main__][INFO] - Epoch 132 dev_accuracy: 91.00000
[2023-11-10 14:14:10,360][__main__][INFO] - Epoch 133 dev_accuracy: 89.80000

[2023-11-10 14:14:15,154][__main__][INFO] - Epoch 134 dev_accuracy: 91.80000
[2023-11-10 14:14:19,868][__main__][INFO] - Epoch 135 dev_accuracy: 90.80000
[2023-11-10 14:14:24,525][__main__][INFO] - Epoch 136 dev_accuracy: 91.40000
[2023-11-10 14:14:29,220][__main__][INFO] - Epoch 137 dev_accuracy: 90.40000
[2023-11-10 14:14:33,852][__main__][INFO] - Epoch 138 dev_accuracy: 90.60000
[2023-11-10 14:14:38,516][__main__][INFO] - Epoch 139 dev_accuracy: 91.20000
[2023-11-10 14:14:43,166][__main__][INFO] - Epoch 140 dev_accuracy: 90.60000
[2023-11-10 14:14:47,815][__main__][INFO] - Epoch 141 dev_accuracy: 90.60000
[2023-11-10 14:14:52,409][__main__][INFO] - Epoch 142 dev_accuracy: 90.60000
[2023-11-10 14:14:57,041][__main__][INFO] - Epoch 143 dev_accuracy: 90.60000
[2023-11-10 14:15:01,572][__main__][INFO] - Epoch 144 dev_accuracy: 91.60000
[2023-11-10 14:15:06,072][__main__][INFO] - Epoch 145 dev_accuracy: 90.60000
[2023-11-10 14:15:10,659][__main__][INFO] - Epoch 146 dev_accuracy: 91.00000
[2023-11-10 14:15:15,275][__main__][INFO] - Epoch 147 dev_accuracy: 90.00000
[2023-11-10 14:15:19,871][__main__][INFO] - Epoch 148 dev_accuracy: 90.40000
[2023-11-10 14:15:24,580][__main__][INFO] - Epoch 149 dev_accuracy: 89.40000
[2023-11-10 14:15:29,324][__main__][INFO] - Epoch 150 dev_accuracy: 89.20000
[2023-11-10 14:15:34,050][__main__][INFO] - Epoch 151 dev_accuracy: 90.00000
[2023-11-10 14:15:38,840][__main__][INFO] - Epoch 152 dev_accuracy: 90.40000
[2023-11-10 14:15:43,615][__main__][INFO] - Epoch 153 dev_accuracy: 90.60000
[2023-11-10 14:15:48,278][__main__][INFO] - Epoch 154 dev_accuracy: 90.60000
[2023-11-10 14:15:53,060][__main__][INFO] - Epoch 155 dev_accuracy: 90.60000
[2023-11-10 14:15:57,631][__main__][INFO] - Epoch 156 dev_accuracy: 92.00000
[2023-11-10 14:16:02,291][__main__][INFO] - Epoch 157 dev_accuracy: 91.60000
[2023-11-10 14:16:06,840][__main__][INFO] - Epoch 158 dev_accuracy: 90.00000
[2023-11-10 14:16:11,485][__main__][INFO] - Epoch 159 dev_accuracy: 92.60000
[2023-11-10 14:16:18,463][__main__][INFO] - Epoch 160 dev_accuracy: 92.20000
[2023-11-10 14:16:22,978][__main__][INFO] - Epoch 161 dev_accuracy: 90.80000
[2023-11-10 14:16:27,660][__main__][INFO] - Epoch 162 dev_accuracy: 91.00000
[2023-11-10 14:16:32,334][__main__][INFO] - Epoch 163 dev_accuracy: 91.00000
[2023-11-10 14:16:37,013][__main__][INFO] - Epoch 164 dev_accuracy: 91.60000
[2023-11-10 14:16:41,691][__main__][INFO] - Epoch 165 dev_accuracy: 91.40000
[2023-11-10 14:16:46,202][__main__][INFO] - Epoch 166 dev_accuracy: 92.40000
[2023-11-10 14:16:50,837][__main__][INFO] - Epoch 167 dev_accuracy: 91.80000
[2023-11-10 14:16:55,535][__main__][INFO] - Epoch 168 dev_accuracy: 91.00000
[2023-11-10 14:17:00,191][__main__][INFO] - Epoch 169 dev_accuracy: 90.80000
[2023-11-10 14:17:04,970][__main__][INFO] - Epoch 170 dev_accuracy: 91.20000
[2023-11-10 14:17:09,709][__main__][INFO] - Epoch 171 dev_accuracy: 90.40000
[2023-11-10 14:17:14,279][__main__][INFO] - Epoch 172 dev_accuracy: 91.40000
[2023-11-10 14:17:19,012][__main__][INFO] - Epoch 173 dev_accuracy: 91.40000
[2023-11-10 14:17:23,564][__main__][INFO] - Epoch 174 dev_accuracy: 92.00000
[2023-11-10 14:17:28,194][__main__][INFO] - Epoch 175 dev_accuracy: 91.40000
[2023-11-10 14:17:32,889][__main__][INFO] - Epoch 176 dev_accuracy: 91.60000
[2023-11-10 14:17:37,526][__main__][INFO] - Epoch 177 dev_accuracy: 92.20000
[2023-11-10 14:17:42,246][__main__][INFO] - Epoch 178 dev_accuracy: 91.20000
[2023-11-10 14:17:46,985][__main__][INFO] - Epoch 179 dev_accuracy: 90.80000
[2023-11-10 14:17:51,663][__main__][INFO] - Epoch 180 dev_accuracy: 90.00000
[2023-11-10 14:17:56,310][__main__][INFO] - Epoch 181 dev_accuracy: 91.40000
[2023-11-10 14:18:00,944][__main__][INFO] - Epoch 182 dev_accuracy: 92.40000
[2023-11-10 14:18:05,651][__main__][INFO] - Epoch 183 dev_accuracy: 91.60000
[2023-11-10 14:18:10,295][__main__][INFO] - Epoch 184 dev_accuracy: 91.80000
[2023-11-10 14:18:15,092][__main__][INFO] - Epoch 185 dev_accuracy: 90.40000
[2023-11-10 14:18:19,698][__main__][INFO] - Epoch 186 dev_accuracy: 92.40000
[2023-11-10 14:18:24,300][__main__][INFO] - Epoch 187 dev_accuracy: 90.80000

[2023-11-10 14:18:28,950][__main__][INFO] - Epoch 188 dev_accuracy: 90.40000
[2023-11-10 14:18:33,594][__main__][INFO] - Epoch 189 dev_accuracy: 91.20000
[2023-11-10 14:18:38,350][__main__][INFO] - Epoch 190 dev_accuracy: 91.20000
[2023-11-10 14:18:42,952][__main__][INFO] - Epoch 191 dev_accuracy: 91.60000
[2023-11-10 14:18:47,606][__main__][INFO] - Epoch 192 dev_accuracy: 90.60000
[2023-11-10 14:18:52,385][__main__][INFO] - Epoch 193 dev_accuracy: 90.80000
[2023-11-10 14:18:57,011][__main__][INFO] - Epoch 194 dev_accuracy: 91.40000
[2023-11-10 14:19:01,717][__main__][INFO] - Epoch 195 dev_accuracy: 90.80000
[2023-11-10 14:19:06,364][__main__][INFO] - Epoch 196 dev_accuracy: 90.20000
[2023-11-10 14:19:11,106][__main__][INFO] - Epoch 197 dev_accuracy: 90.80000
[2023-11-10 14:19:15,880][__main__][INFO] - Epoch 198 dev_accuracy: 91.40000
[2023-11-10 14:19:20,441][__main__][INFO] - Epoch 199 dev_accuracy: 90.00000
[2023-11-10 14:19:25,121][__main__][INFO] - Epoch 200 dev_accuracy: 90.60000
[2023-11-10 14:19:29,848][__main__][INFO] - Epoch 201 dev_accuracy: 89.80000
[2023-11-10 14:19:34,524][__main__][INFO] - Epoch 202 dev_accuracy: 90.80000
[2023-11-10 14:19:39,236][__main__][INFO] - Epoch 203 dev_accuracy: 90.60000
[2023-11-10 14:19:43,848][__main__][INFO] - Epoch 204 dev_accuracy: 91.00000
[2023-11-10 14:19:48,588][__main__][INFO] - Epoch 205 dev_accuracy: 91.80000
[2023-11-10 14:19:53,214][__main__][INFO] - Epoch 206 dev_accuracy: 91.20000
[2023-11-10 14:19:57,906][__main__][INFO] - Epoch 207 dev_accuracy: 91.40000
[2023-11-10 14:20:02,568][__main__][INFO] - Epoch 208 dev_accuracy: 91.00000
[2023-11-10 14:20:07,195][__main__][INFO] - Epoch 209 dev_accuracy: 91.60000
[2023-11-10 14:20:11,758][__main__][INFO] - Epoch 210 dev_accuracy: 91.60000
[2023-11-10 14:20:16,440][__main__][INFO] - Epoch 211 dev_accuracy: 91.80000
[2023-11-10 14:20:21,036][__main__][INFO] - Epoch 212 dev_accuracy: 90.80000
[2023-11-10 14:20:25,600][__main__][INFO] - Epoch 213 dev_accuracy: 91.60000
[2023-11-10 14:20:30,138][__main__][INFO] - Epoch 214 dev_accuracy: 91.60000
[2023-11-10 14:20:34,760][__main__][INFO] - Epoch 215 dev_accuracy: 91.60000
[2023-11-10 14:20:39,579][__main__][INFO] - Epoch 216 dev_accuracy: 91.40000
[2023-11-10 14:20:44,225][__main__][INFO] - Epoch 217 dev_accuracy: 91.00000
[2023-11-10 14:20:48,782][__main__][INFO] - Epoch 218 dev_accuracy: 90.40000
[2023-11-10 14:20:53,414][__main__][INFO] - Epoch 219 dev_accuracy: 91.20000
[2023-11-10 14:20:58,048][__main__][INFO] - Epoch 220 dev_accuracy: 91.60000
[2023-11-10 14:21:02,669][__main__][INFO] - Epoch 221 dev_accuracy: 91.40000
[2023-11-10 14:21:07,355][__main__][INFO] - Epoch 222 dev_accuracy: 91.60000
[2023-11-10 14:21:12,070][__main__][INFO] - Epoch 223 dev_accuracy: 90.60000
[2023-11-10 14:21:16,635][__main__][INFO] - Epoch 224 dev_accuracy: 90.80000
[2023-11-10 14:21:21,261][__main__][INFO] - Epoch 225 dev_accuracy: 90.40000
[2023-11-10 14:21:25,932][__main__][INFO] - Epoch 226 dev_accuracy: 90.20000
[2023-11-10 14:21:30,566][__main__][INFO] - Epoch 227 dev_accuracy: 91.60000
[2023-11-10 14:21:35,263][__main__][INFO] - Epoch 228 dev_accuracy: 90.20000
[2023-11-10 14:21:40,023][__main__][INFO] - Epoch 229 dev_accuracy: 91.40000
[2023-11-10 14:21:44,684][__main__][INFO] - Epoch 230 dev_accuracy: 90.60000
[2023-11-10 14:21:49,372][__main__][INFO] - Epoch 231 dev_accuracy: 91.00000
[2023-11-10 14:21:54,136][__main__][INFO] - Epoch 232 dev_accuracy: 90.80000
[2023-11-10 14:21:58,935][__main__][INFO] - Epoch 233 dev_accuracy: 91.00000
[2023-11-10 14:22:03,698][__main__][INFO] - Epoch 234 dev_accuracy: 91.80000
[2023-11-10 14:22:08,357][__main__][INFO] - Epoch 235 dev_accuracy: 91.00000
[2023-11-10 14:22:13,054][__main__][INFO] - Epoch 236 dev_accuracy: 91.60000
[2023-11-10 14:22:17,782][__main__][INFO] - Epoch 237 dev_accuracy: 91.20000
[2023-11-10 14:22:22,373][__main__][INFO] - Epoch 238 dev_accuracy: 91.60000
[2023-11-10 14:22:27,085][__main__][INFO] - Epoch 239 dev_accuracy: 92.00000
[2023-11-10 14:22:31,703][__main__][INFO] - Epoch 240 dev_accuracy: 92.80000