NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Assignment for SC4002 / CE4045 / CZ4045

AY2023-2024

Group ID: 21

Group members:

Name	Matric No.
Jiang Jiaxi	U2022209K
Yang Yida	U2022689D
Lim Zhi Qing	U2021897L
Jared Chan	U2022954B
Mak Qin Xiang	U2022775F

Contribution:

Name	Contribution
Jiang Jiaxi	Part 1 code & report
Yang Yida	Part 1 code & report
Lim Zhi Qing	Part 2 code & report
Jared Chan	Part 2 code & report
Mak Qin Xiang	Part 2 code & report

Table of Contents

1. Sequence Tagging: NER

1.1 Word Embedding

Word2vec embeddings provide method “most_similar” to give the most similar words with their respective cosine similarities.
Most similar word to “student”: “students” (cosine similarity: 0.729)
Most similar word to “Apple”: “Apple_AAPL” (cosine similarity: 0.746)
Most similar word to “apple”: “apples” (cosine similarity: 0.720)

1.2 Data

(a) Dataset Description

We processed the files for CoNLL2003, put sentences into arrays which are separated by ‘\n’ in origin files. The sentences in train file and their respective tags are shown in the following table.

	sequences_word	sequences_label
0	[EU, rejects, German, call, to, boycott, Briti...	[I-ORG, O, I-MISC, O, O, O, I-MISC, O, O]
1	[Peter, Blackburn]	[I-PER, I-PER]
2	[BRUSSELS, 1996-08-22]	[I-LOC, O]
3	[The, European, Commission, said, on, Thursday...	[O, I-ORG, I-ORG, O, O, O, O, O, O, I-MISC, O,...
4	[Germany, 's, representative, to, the, Europea...	[I-LOC, O, O, O, O, I-ORG, I-ORG, O, O, O, I-P...
...	...	...
14981	[on, Friday, :]	[O, O, O]
14982	[Division, two]	[O, O]
14983	[Plymouth, 2, Preston, 1]	[I-ORG, O, I-ORG, O]
14984	[Division, three]	[O, O]
14985	[Swansea, 1, Lincoln, 2]	[I-ORG, O, I-ORG, O]

14986 rows × 2 columns

Table 1. Data frame for train dataset

Looking through the dataset, we found there are some rows with [-DOCSTART-], which should be treated as sentences, so we removed these rows and obtained new dataset size.

File	Size (number of sentences)	Size (remove -DOCSTART-)
Training file	14986	14041
Development file	3465	3250
Test file	3683	3453

Table 2. size of dataset files

We choose the tagging schema IOB1. In this schema, I is a token inside a Name Entity chunk, O is a token outside a chunk and B is the beginning of a chunk immediately following another chunk of the same Named Entity.

All possible word labels: O, I-ORG, I-PER, I-LOC, I-MISC, B-ORG, B-PER, B-LOC, B-MISC. (ORG: organisation, PER: person, LOC: location, MISC: Miscellaneous entities, e.g., events, nationalities, products, or works of art.)

(b) Dataset Details

Example sentence: In San Francisco, Pedro Martinez allowed two hits in eight innings and David Segui drove in two runs as the Montreal Expos shut out the San Francisco Giants 3-0 for their third straight wins.

Tags: ['O', 'I-LOC', 'I-LOC', 'O', 'I-PER', 'I-PER', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'I-PER', 'I-PER', 'O', 'O', 'O', 'O', 'O', 'O', 'I-ORG', 'I-ORG', 'O', 'O', 'O', 'I-ORG', 'I-ORG', 'I-ORG', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

Since the dataset is also in IOB1 schema, the label for each word is already the complete named entities. All the named entities in this example sentence: O, I-LOC, I-PER, I-ORG.

1.3 Model

(a) Unknown words

We deal with new words in the training set using random vectors. When converting words into vectors with word2vec embedding, if the word is not found, we use a random vector to represent it. The vector will be in shape (300,) and range from -1 to 1, which is the same as word2vec embeddings. These unknown words are processed and learned by the model as noises.
We went through other NER tagging works, the unknown and padding words usually have their own embeddings. For example, when converting words to their respective IDs, there are two IDs left for UNK and PAD. However, in our case, the word embeddings are not allowed to be updated (frozen), so we choose to use random vectors.

```
def generate_random_vector():
    return np.random.uniform(-1, 1, 300)  # 300 random floats in the range [-1, 1]

def get_word_vector(word):
    try:
        return word2vec[word]
    except KeyError:
        # deal with unknown words
        return generate_random_vector()

df_train['sequences_wordvec'] = df_train['clear_words'].apply(
    lambda x: [get_word_vector(word) for word in x]
)
```

Figure 1. Code snippet dealing with unknowns.

(b) Neural Networks - LSTM

The overall network architecture is shown in figure 2. We used a bidirectional LSTM model to explore different parameters' affection on the model performance. We carried out experiments with various LSTM units, dropout values, learning rate and batch sizes. The following part states the statistics of the model gives the best result. The mathematical functions used for the forward computation is the same as the one in part 2 for the portion of bidirectional LSTM layer and the softmax layer. And it is clearly stated in part 2 Question 2 (c).

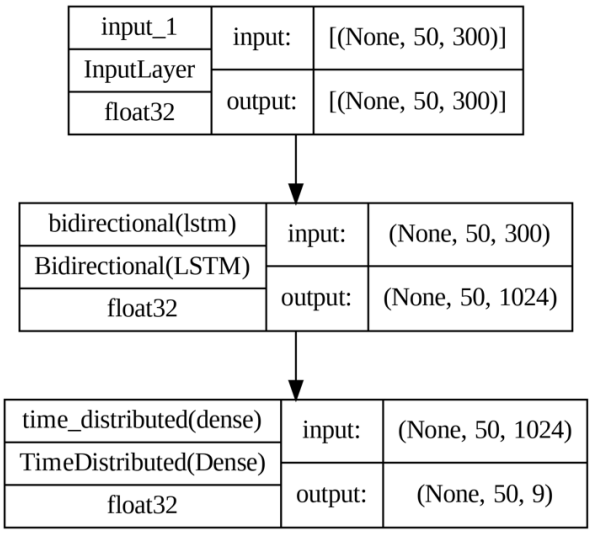

Figure 2. Neural network architecture.

Bidirectional LSTM Layer

It takes a 50 * 300 vector as input. 50 represents the number of words per sentence. 300 is the word vector of each word. The unit parameter 512 in this layer specifies the dimensionality of the output space of the LSTM layer. This parameter determines the number of memory units or cells in the LSTM layer. Each unit or cell is responsible for learning and remembering patterns in the input data. The more units a model has, the more complex patterns the LSTM layer can capture. However, a higher number of units also means a higher computational cost, and it may lead to overfitting if the model has insufficient data or if the model becomes too complex for the given problem. return_sequences=True means that the layer returns the full sequence of outputs for each input sequence, rather than just the output at the last time step. recurrent_dropout is set to 0.1, representing the dropout applied to the recurrent connections during training. In total, according to figure 3, the Bi-directional LSTM layer has 3330048 trainable parameters. The weights and biases being updated can be seen in figure 4.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional_1 (Bidirecti  (None, 50, 1024)          3330048
 onal)

 time_distributed_1 (TimeDi  (None, 50, 9)             9225
 stributed)

=================================================================
Total params: 3339273 (12.74 MB)
Trainable params: 3339273 (12.74 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3. Model summary

Output Layer

This layer applies a dense (fully connected) neural network to each time step of the sequence independently. It takes the final representation of each word, which is a vector with dimension of 1024 as input, and outputs the final result by going through a softmax layer. In total, according to figure 3, the Bi-directional LSTM layer has 9225 trainable parameters. The weights and biases being updated can be seen in figure 4.

Layer	Type	Shape
bidirectional_1	Weight	(300, 2048)
bidirectional_1	Weight	(512, 2048)
bidirectional_1	Bias	(2048,)
bidirectional_1	Weight	(300, 2048)
bidirectional_1	Weight	(512, 2048)
bidirectional_1	Bias	(2048,)
time_distributed_1	Weight	(1024, 9)
time_distributed_1	Bias	(9,)

Figure 4. Parameters updated at each layer.

(c) Epochs used

We used 7 epochs. The running time is 1428.36 seconds.

(d) Experiment Results

The f1 score on the test set is 0.787. The f1 score of development set is shown in Figure 5.

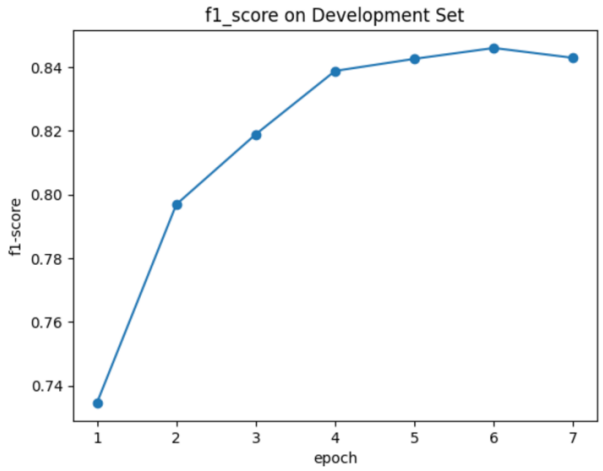

Figure 5. f1_score on development set

2. Sentence-Level Categorization: Question Classification

(a) Classes Used

The 5 classes used are 1, 2, 3, 4 and OTHERS.

(b) Aggregation Methods

We tested global max pooling and global average pooling as our aggregation methods. We adopted the global max pooling as it gave us a better accuracy as compared to global average pooling.

(c) Neural Network Architecture

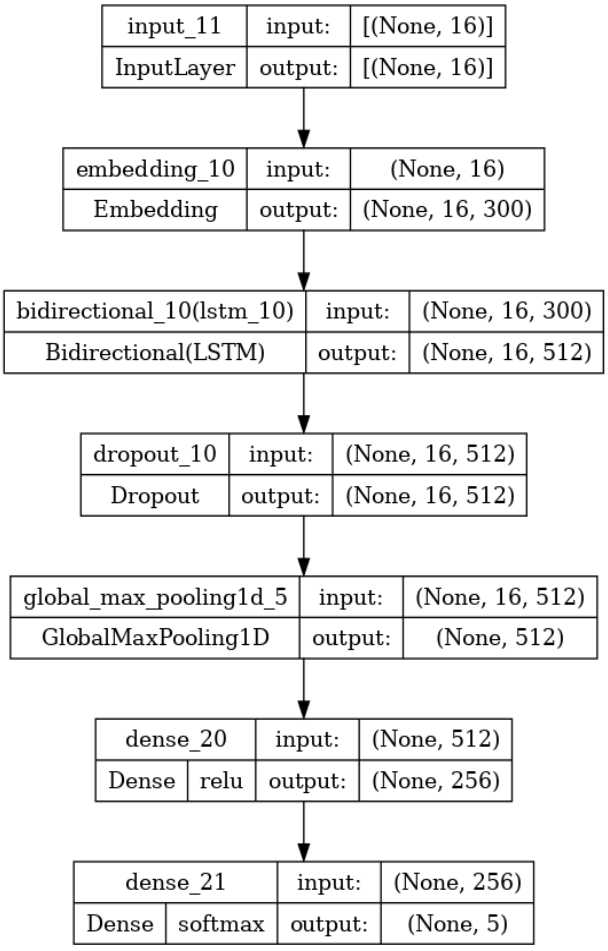

Figure 6. Neural network architecture

Layer	Parameter	Shape
bidirectional (forward)	Weight	(300, 1024)
bidirectional (forward)	Bias	(1024,)
bidirectional (backwards)	Weight	(256, 1024)
bidirectional (backwards)	Bias	(1024,)
dense	Weight	(512, 256)
dense	Bias	(256,)
output	Weight	(256, 5)
output	Bias	(5,)

Figure 7. Parameters updated at each layer.

Embedding Layer

The first layer in the model is the embedding layer. The input is a 16 word sequence of tokenized text. Each word is converted into word embeddings by indexing the pre-trained Word2Vec embedding matrix. The resulting output is a sequence of 16 words, each with 300 features, giving the embedding layer 2400600 untrained parameters.

Bi-directional LSTM Layer

The bi-directional LSTM layer of 256 cells processes the embedded sequences in both the forwards and backwards directions, capturing dependencies from both past and future contexts. The output is a tensor of sequence length 16 with hidden dimensions 512. The dimensions of the hidden states is double the number of LSTM cells due to the bidirectional nature of the LSTM. In total, the Bi-directional LSTM layer has 1140736 trainable parameters. The weights and biases being updated can be seen in figure 7.

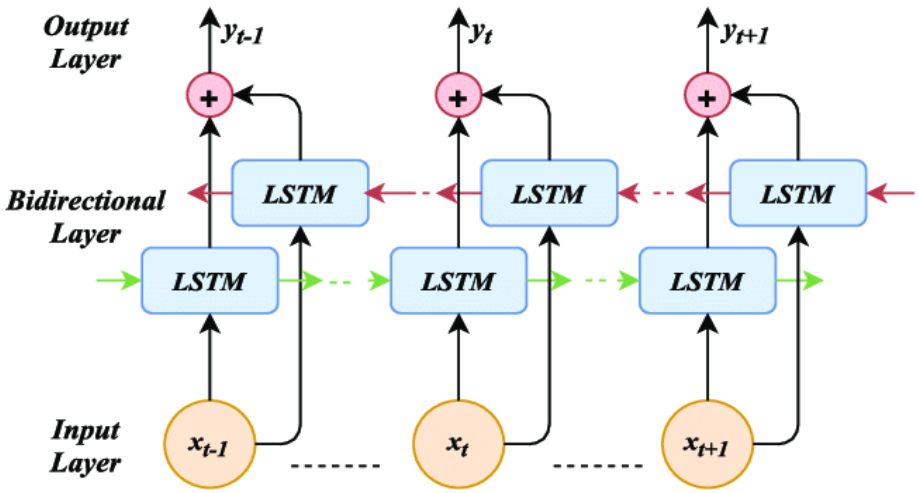

Figure 8. Bi-directional LSTM Model [1].

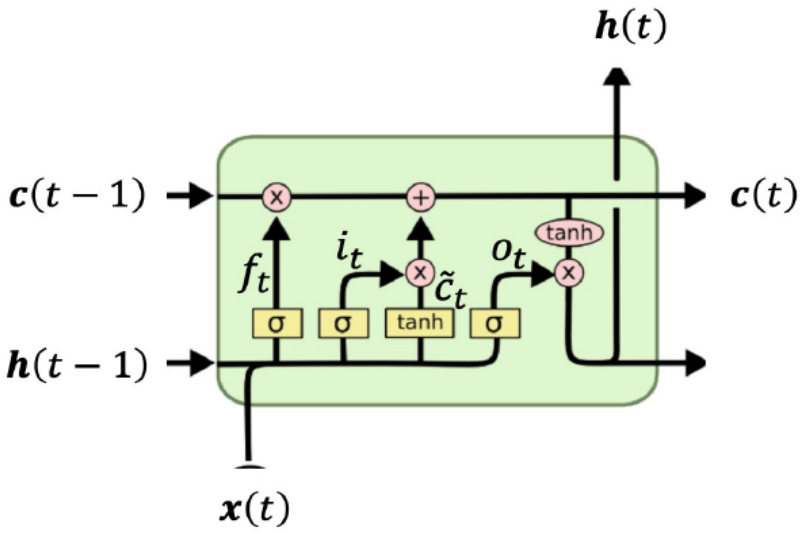

Figure 9. LSTM cell

In LSTM networks, the core idea revolves around the cell state, which allows information to flow along it with some controlled modifications through gates. LSTM cells have three main gates, which are the forget gate, input gate and output gate. The following paragraphs on LSTM were taken from the CZ4042 Neural Network lecture notes on RNN.

The forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed. It is the f(t) as shown in figure 3, and it determines if c(t-1) is to be remembered or not.

Mathematical function: $\boldsymbol{f}(t) = \sigma\left(\boldsymbol{U}_{\boldsymbol{f}}^{\top}\boldsymbol{x}(t) + \boldsymbol{W}_{\boldsymbol{f}}^{\top}\boldsymbol{h}(t-1) + \boldsymbol{b}_{\boldsymbol{f}}\right)$

The input gate can allow incoming signals to alter the state of the memory cell or block it. It decides what new information to store in the cell stage. This has two parts: A sigmoid input gate layer decides which values to update and a tanh layer that creates a vector of new candidate values c̃(t) that could be added to the state.

Mathematical functions: $\boldsymbol{i}(t) = \sigma\left(\boldsymbol{U}_{\boldsymbol{i}}^{\top}\boldsymbol{x}(t) + \boldsymbol{W}_{\boldsymbol{i}}^{\top}\boldsymbol{h}(t-1) + \boldsymbol{b}_{\boldsymbol{i}}\right)$

$$\tilde{\boldsymbol{c}}(t) = \phi\left(\boldsymbol{U}_{\boldsymbol{c}}^{\top}\boldsymbol{x}(t) + \boldsymbol{W}_{\boldsymbol{c}}^{\top}\boldsymbol{h}(t-1) + \boldsymbol{b}_{\boldsymbol{c}}\right)$$

Cell state mathematical function: $\boldsymbol{c}(t) = \tilde{\boldsymbol{c}}(t) \odot \boldsymbol{i}(t) + \boldsymbol{c}(t-1) \odot \boldsymbol{f}(t)$

The output gate can allow the state of the memory cell to have an effect on other neurons or prevent it.

Mathematical functions: $\boldsymbol{o}(t) = \sigma\left(\boldsymbol{U}_{\boldsymbol{o}}^{\top}\boldsymbol{x}(t) + \boldsymbol{W}_{\boldsymbol{o}}^{\top}\boldsymbol{h}(t-1) + \boldsymbol{b}_{\boldsymbol{o}}\right)$

$$\boldsymbol{h}(t) = \phi\left(\boldsymbol{c}(t)\right) \odot \boldsymbol{o}(t)$$

Dropout Layer

The dropout layer prevents overfitting during training by randomly setting a fraction of input units to zero, helping with regularisation and generalisation of the model.

Pooling Layer

The pooling layer performs global max pooling on the output of the dropout layer, computing the maximum value of each feature over the sequence length, capturing the most important features from the input sequence. This layer takes in an input of dimension (16, 512) and reduces it to an output of dimension (512).

Densing Layer

The dense layer is a fully connected layer that takes the output of the pooling layer as input. The ReLU (Rectified Linear Unit) activation function is applied element-wise to each neuron's

output. Negative values are replaced with zero while positive values are unchanged. This layer consists of 131328 trainable parameters, and the weights and biases being updated can be seen in figure 7. The resulting output is of length 256, which is the number of neurons in the dense layer. This output is the final vector representation of all the words in the input text, which is fed to the softmax classifier in the output layer.

Output Layer

Finally, the output layer is another dense layer consisting of 1285 trainable parameters, and the weights and biases being updated can be seen in figure 7. This layer contains 5 neurons, equal to the number of classification labels. Each neuron represents a class label. The softmax activation function takes the vector output of the previous layer as input and produces a probability distribution over the classes. Each class is assigned a probability and the class with the highest probability is predicted as the model's output.

(d) Epochs Used

In total the training took 7.6781 seconds, with 8 epochs.

(e) Accuracy

The accuracy on the development set for each training epoch are as follows.

Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	0.1982	0.9376	0.3981	0.856
2	0.1236	0.9548	0.3691	0.884
3	0.0461	0.9838	0.3638	0.900
4	0.0165	0.9966	0.4411	0.892
5	0.0079	0.9978	0.4575	0.882
6	0.0137	0.9956	0.4784	0.884
7	0.0058	0.9986	0.4671	0.890
8	0.0233	0.9919	0.5868	0.850

Table 3. Accuracy on Training and Validation Set

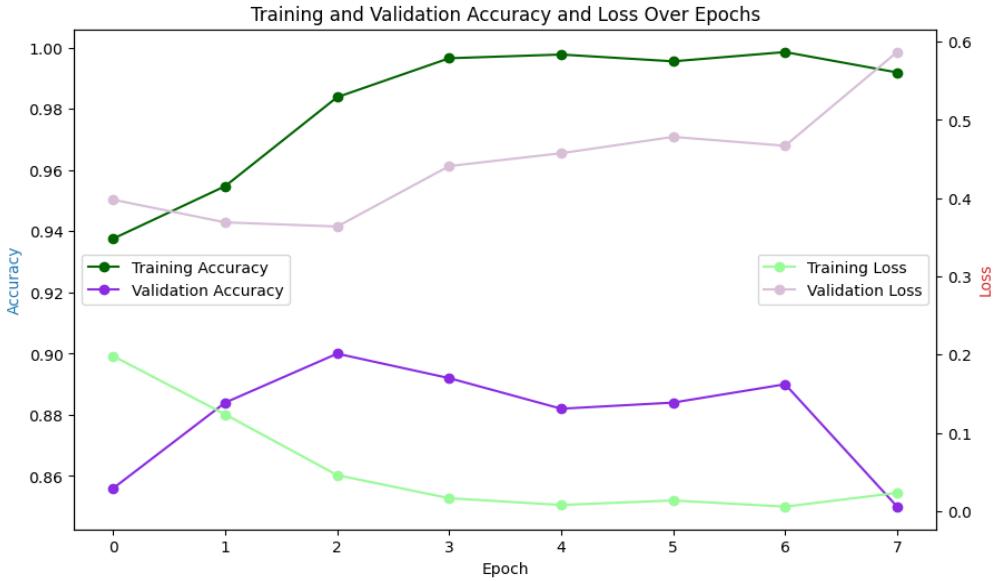

Figure 10. Graph of Accuracy on Training and Validation Set

The accuracy on the test set is 0.9040.

References

[1] I. K. Ihianle, A. O. Nwajana, S. H. Ebenuwa, R. I. Otuka, K. Owa, and M. O. Orisatoki, “A deep learning approach for human activities recognition from multimodal sensing devices,” IEEE Access, vol. 8, pp. 179028–179038, Jan. 2020, doi: 10.1109/access.2020.3027979.