CS570 Application of Deep Learning

Assignment 2: CNN and RNN

1 Objective

This assignment aims to help you gain hands-on experience with Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). The key learning objectives include:

1. **Implementing CNN and RNN**: Practice designing and training a CNN and an RNN model using a deep learning library like Keras.
2. **Applying CNN and RNN**: Apply CNN and RNN models to effectively solve a real-world Natural Language Processing (NLP) problem.
3. **Comparing word embedding approaches**: Understand the difference between training your own word embeddings and using pre-trained ones.

2 Tasks

In this assignment, you will implement and compare Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for classifying news articles from the AG News Classification dataset. You will also explore how different word embeddings perform in capturing patterns in text data and justify your findings.

2.1 Dataset preprocessing

1. (20 points) Load the dataset and write Python code to do the following.
   (a) Extract the news *Description* from the dataset to use as input text data.
   (b) Tokenize the training text data to build the vocabulary and convert text to sequences. Use maximum vocabulary size to be 10000. Use symbol `<UNK>` for words not in the vocabulary.
   (c) Pad the sequences to a fixed length of 150. Add padding to the end of the sequences and truncate from the end.
   (d) Extract the *ClassIndex* from the dataset and transform the labels to be zero-based (i.e., starting from 0 rather than 1).

2.2 Training a CNN and an RNN model

2. (50 points) Write Python code to do the following.
   (a) Design a 1D CNN for news classification. The model should include two `Conv1D` layers, each with 32 filters and a kernel size of 3. Apply a `GlobalMaxPooling1D` layer to downsample the feature maps. Follow this with a `Dense` layer containing 64 neurons and a dropout rate of 20%. Add a final output layer with a softmax activation function.
   (b) Train the CNN model using `sparse_categorical_crossentropy` loss function and `adam` optimizer for 10 epochs and setting batch size to 128. Generate word embeddings of size 100 based on the training text data and use them as input features for training the model. Evaluate the CNN model on the test dataset using the `accuracy` metric.
   (c) Design an LSTM model (a RNN variant) for news classification. The model should include one `LSTM` layer with the number of units or hidden states set to 64. Apply two `Dense` layers: the first with 128 neurons followed by a 25% dropout and the second with 64 neurons followed by a 10% dropout. Add a final output layer with a softmax activation function.

(d) Train the LSTM model using `sparse_categorical_crossentropy` loss function and `adam` optimizer for 10 epochs and setting batch size to 128. Use the word embeddings generated based on the training text data as input features for training the model. Evaluate the LSTM model on the test dataset using the `accuracy` metric.

(e) Compare the CNN and LSTM models in terms of accuracy and training time. Based on your experiment, which model did you find suitable for the classification task and why?

2.3 Applying pre-trained word embeddings

3. (30 points) Write Python code to do the following.

(a) Load pre-trained Word2Vec embeddings using the **gensim** library and use them to initialize the embeddings for the words in the AG News dataset vocabulary.

(b) Train the same CNN and LSTM model architectures for news classification and evaluate on the test dataset.

(c) Analyze and report how the change in word embeddings impacts the performance (*i.e.*, accuracy) of your models. Explain your observations for each case.

3 Submission Guidelines

- **Allowed Libraries:**

  You are only permitted to use **Keras**, **pandas**, **gensim**, and **scikit-learn** libraries. No other libraries are allowed.

- **Python Version:**

  Ensure your code is compatible with Python 3.8 or higher.

- **Code Comments and Answers:**

  - Include proper comments in your code to explain key steps.
  - Use TEXT cells in the notebook to provide answers to questions. Do not create separate documents for answers.
  - Use TEXT cells to separate the code for each section.

- **Organization:**

  Consolidate all your Python code into a **single** Jupyter Notebook. Do not create multiple notebooks for different sections.

- **Submission:**

  Name your notebook as *CS570-assignment-2-<your-first-name>.ipynb* and submit it through Blackboard.