Assignment 8

TDT4171 — Artificial Intelligence Methods

March 2024

Information

- **Delivery deadline: March 17, 2024 by 23:59**. No late delivery will be graded! Deadline extensions will only be considered for extraordinary situations such as family or health-related circumstances. These circumstances must be documented, e.g., with a doctor's note ("legeerklæring"). Having a lot of work in other classes is not a legitimate excuse for late delivery.

- Cribbing("koking") from other students is not accepted, and if detected, will lead to immediate failure of the course. The consequence will apply to both the source and the one cribbing.

- Students can **not** work in groups. Each student can only submit a solution individually.

- Required reading for this assignment: Chapter 22. Deep Learning (the parts in the curriculum found on Blackboard "Sources and syllabus" → "Preliminary syllabus") of Artificial Intelligence: A Modern Approach, Global Edition, 4th edition, Russell & Norvig.

- For help and questions related to the assignment, **ask the student assistants during the guidance hours**. The timetable for guidance hours can be found under "Course work" on Blackboard. For other inquires, an email can be sent to tdt4171@idi.ntnu.no.

- Deliver your solution on Blackboard. Please upload your assignment as one PDF report and one source file containing the code (i.e., one .py file) as shown in Figure 1.

Figure 1: Delivery Example

Assignment Information

In this assignment, you will try out a feedforward neural network and a recurrent neural network on a dataset containing text-based reviews of restaurants. The original dataset is available from https://www.yelp.com/dataset, but we will work on a preprocessed version available on Blackboard. The goal of the assignment is to be able to learn the meaning of each review: Is it positive or negative? You will not need to do the detailed implementations yourself. Instead, we will use the Python package keras. If you are unfamiliar with how to install packages in Python, you may want to take a look here.

TensorFlow is a library for deep learning. While flexible and powerful, it can be seen as having a steep learning curve. We will therefore use keras as a simplifying abstraction layer in this assignment. Keras is part of TensorFlow, and thus TensorFlow needs to be installed. The installation guide for TensorFlow is available from here, and the getting started guide for keras is available from here.

A Python file (assignment_8.py) containing helper functions is available at Blackboard. You may use this file as a starting point for your implementation. This code and the instructions given in this document has been tested on Python 3.8, Tensorflow 2.9.

About the data

Use the data in the file `keras-data.pickle` available on Blackboard. The data is stored in a pickle-file containing a dictionary. To load the data, use the following Python code:

```python
import pickle

with open(file="keras-data.pickle", mode="rb") as file:
    data = pickle.load(file)
```

This gives you a dictionary with 6 keys:

- `x_train`: A list of encoded reviews. The data is encoded as lists of word-id's, e.g. [117, 143, 307, 468, 38, 117, 411, ...]. This is the data we will use for training the classifiers.

- `y_train`: A list of integers. This is the target of the learning, with 1 representing a positive review and 0 representing a negative one. The ordering of elements fit that of `x_train`.

- `x_test`: A list of encoded reviews similar to `x_train`. This is the test-set, and should not be used during training.

- `y_test`: A list of integers. These are the class-labels for x_test and are used at test-time.

- `vocab_size`: Holds information about the number of different words in the vocabulary

- `max_length`: Gives the longest review in the dataset (in terms of the number of words used).

In this assignment, we will build neural networks, and this requires first making each sequence (that is, each document) have the same length. Use `pad_sequences` from `tf.keras.utils` to accomplish this. Both the training-data and the test-data must be padded to the same length; play around with the maxlen parameter to trade off speed and accuracy.

Exercise

To build the feedforward neural network and the recurrent neural networks, we will use the `tf.keras.Sequential` API. After defining a model by writing `model = tf.keras.Sequential()`, we can build up the model by simply adding layers one by one.

Both models should start with an `tf.keras.layers.Embedding`, that embeds the word-vectors into a high-dim metric space (`model.add(tf.keras.layers.Embedding(<parameters>)))` will do that for you; take care when choosing the parameters).

The feedforward neural network should be build using `tf.keras.layers.Dense` layers. You can flatten the embedding by including a `tf.keras.layers.Flatten` layer.

For the recurrent neural network you need a `tf.keras.layers.LSTM` layer and a `tf.keras.layers.Dense` layer at the end. Take care when you choose the parameters as well for these layers.

For a model, you can use `model.fit()` to train it. The dataset is huge, so if you run this on your own computer, you may want to start with one epoch (the number of epochs can be chosen by setting the `epochs:int` argument in the call to `model.fit`).

Finally, evaluate the model with `model.evaluate(x_test, y_test)`.

For both models, you should try to reach an accuracy of at least 90% on the test data.

Deliver a PDF that documents the parameters you have used when defining the models, and the obtained results, along with your code. Make sure that the code is runnable without any modifications after delivery. In addition, the code must be human-readable and contain explaining comments where appropriate. Commenting is especially important if the code does not work.