# Deep Learning for NLP 2022
# Homework 3

**Due on May 30, 2022 at 23:59**

TECHNISCHE UNIVERSITÄT DARMSTADT

Version of May 23, 2022

Overall: 11 points.
This homework involves training more complex neural networks which can take some time.
**Plan accordingly for the extended training time!**

**Submission Guidelines**

This homework comes with a zip archive containing the data and code templates. Write your code only in the `run.py` files. Then upload a zip file with the same folder structure, i.e.:

```
hw3_group###.zip
 |__p1/
 |   |__run.py
 |   |__data-dev.txt
 |   |__data-test.txt
 |   |__data-train.txt
 |__p2/
 |   |__run.py
 |   |__data.txt
 |__hw3.pdf
```

- `hw3.pdf` should answer the questions **and** report your training results.
- **Do NOT** include the FastText embeddings, but expect them to be available at `p1/wiki-news-300d-1M.vec`.
- `###` is your group number (e.g., the file name could be `hw3_group2.zip`).
- Please make sure that your code prints **all** relevant results for **all** subtasks when the tutors run `run.py`.[a]

Make sure that the results you report are reproducible and that your code is runnable in the conda environment for this course. If you are aware that your network never stops training, please be honest and add a short statement saying so. Thank you!

[a] If you think that it makes your code more readable/structured, you can also submit other `.py` files whose functions are called in `p1.py`. But the tutors will only execute `p1.py`.

## 1 Getting to Know Keras: Semantic Textual Similarity (7P)

In this task, we define *semantic textual similarity* (STS) as a **supervised** regression task in which the semantic similarity of two pieces of text (typically sentences) should be determined.

## 1.1 Data Formats (1P)

The labeled data sets for this task contain entries that comprise a real-numbered similarity score between 0 and 1, a first sentence, and a second sentence. The unlabeled data sets' entries comprise only the two sentences. In our case, each line in the data sets corresponds to a single entry, and the score and the sentences are separated by tabs `\t`.

- Implement a reader function for the labeled data sets, which for a given filename returns a list of scores, a list of first sentences, and a list of second sentences.

**Report** the first sentence pair in the training set and its similarity score in the PDF and **print** them in your code.

**Hint:** Set the encoding to `utf8` and use `strip()` to remove whitespace from the beginning and ending of the sentences.

## 1.2 Embedding the Sentences (2P)

We will use the averages of the words' FastText[1] embeddings to embed both sentences.

a) Download the `wiki-news-300d-1M.vec.zip` embeddings and read them into a Python dictionary that maps every token to the corresponding vector. Represent the vectors as NumPy arrays. Only load the first 40 000 lines of the file. Your code should use the file `wiki-news-300d-1M.vec` in the `p1/` folder.

b) Implement a function that tokenizes the sentences using `nltk.word_tokenize`[2].

   **Report** the tokenized first sentence from the first training set entry (it starts with "A brown dog") in your PDF and **print** it in your code.

c) Implement a function that maps the tokens to their corresponding embedding vectors. If a token does not exist in FastText's vocabulary, embed this token as a 0-vector with the same dimension as the FastText embeddings.

d) Finally, implement a function that embeds each sentence as the average of the embeddings of its tokens.

   **Report** the first 20 dimensions of the embedding of the same sentence (first sentence from the first training set entry) in your PDF and **print** it in your code.

## 1.3 Scoring the Similarity (4P)

We will train a simple multi-layer perceptron to score the similarity of the two sentences. The MLP should concatenate both inputs and have two hidden layers with dropout. The first hidden layer should have 300 dimensions and ReLu activation, and the second hidden layer (the output layer) should have 1 dimension and sigmoid activation.

For this task, please use the provided Keras module (a high-level neural network library) from tensorflow. If you have trouble getting started, the Keras documentation (`https://www.tensorflow.org/versions/r2.6/api_docs/python/tf`) and online tutorials[3] can be of great help.

The precise definition is as follows:

- Two inputs, one for each sentence embedding.

  **Hint:** As we use two sentences, you can use two `Input` layers and one `Concatenate` layer.

- A layer that concatenates both inputs.

  **Hint:** You may use `tensorflow.keras.layers.Concatenate`.

- A dropout layer with probability 0.3.

  **Hint:** You may use `tensorflow.keras.layers.Dropout`.

[1]`https://fasttext.cc/docs/en/english-vectors.html`

[2]nltk is a basic NLP library. Some examples of the usage of `word_tokenize` can be found here: `https://www.nltk.org/book/ch03.html#accessing-text-from-the-web-and-from-disk`.

[3]E.g., `https://www.tensorflow.org/tutorials/keras/classification`

- A dense layer with 300 dimensions and relu activation.

  **Hint:** You may use `tensorflow.keras.layers.Dense`.
- A dropout layer with probability 0.3.
- A dense layer with 1 dimension and sigmoid activation.

  **Hint:** You may use `tensorflow.keras.Model(<input>,<output>)` to define the model.

**Tasks:**

a) Briefly explain why we are using sigmoid instead of softmax as the activation function in the final layer.

b) Implement the model described above.

   **Print** the model summary using Keras' `summary` method[4].

c) Compile the model with Adam as optimizer and mean squared error as the loss function. Also, choose mean squared error as a metric. (All these are readily provided by tensorflow.keras.)

d) Train the model with batch size 100 for 300 epochs on the prepared training data set. Observe the mean squared error on the development data set.

e) **Report** and **print** the model's final mean squared error on the development data set.

[4]See `https://www.tensorflow.org/api_docs/python/tf/keras/Model#summary`

## 2 Text Classification with CNNs (4P)

In this task, you will perform text classification on the 20 newsgroups dataset. It is a collection of e-mails coming from different newsgroups. The goal is to assign each e-mail to its corresponding newsgroup.

Some tensorflow Keras code snippets are provided for this task. There is not much code to write, but training CNNs is a computationally intensive task! **Plan accordingly for the extended training time.**

*Hint*: You can check out Google Colab for "free" GPU access.

### 2.1 Creating Data Splits (0P)

In `run.py`, you will find code which generates train, dev and test sets for the 20 Newsgroups dataset (`data.txt`).

`train_y` will be a list of 20-component one-hot vectors representing newsgroups, and `train_x` will be a list of 300-component vectors where each entry corresponds to a word ID. Each 300-component vector represents an e-mail.

### 2.2 A basic CNN (1P)

Add a basic CNN to the provided skeleton code. The CNN should feature:

- A convolutional layer with 75 filters and filter size $k = 2$, using a ReLU activation function
- A global max pooling layer
- A softmax output layer

**Print** the model summary using Keras' `summary` method. In the PDF, **report** your accuracy on the dev set after training **print** it in your code.

### 2.3 Early Stopping (3P)

Based on the basic CNN from 2.2, create a new model that uses early stopping to determine the optimal number of epochs. This can be done by setting a high number of epochs (say, 50) initially, recording the best achieved result on the dev set during training, and stopping early if there are no improvements after a set amount of epochs. For this homework, stop early if there are no improvements after 2 epochs.

To implement this in Keras, callbacks can be supplied to the training process. In particular, have a look at `ModelCheckpoint`[5] and `EarlyStopping`[6].

Implement the model with the parameters as described above. Use Keras' `model.load_weights` before evaluating to load the best model checkpoint.

In your PDF, **report** the dev and test accuracy of your best model and **print** them in your code.

[5]Keras `ModelCheckpoint`: `https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint`
[6]Keras `EarlyStopping`: `https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping`