# Deep Learning for NLP 2019
# Home Exercise 04

TECHNISCHE UNIVERSITÄT DARMSTADT

**Due on Tuesday, 14.05. at 13:00**

**Submission Guidelines for all Home Exercises**

- When submitting multiple files, submit one **zip-archive**.
- Submit python code as plain python scripts (**.py**). **Must** be runnable in the given Docker container.
- Submit answers to non-code assignments in **one PDF** file. Scans are permitted, if readable.
- Guidelines specific to neural network code:
  - Please submit your training/testing results (a copy of your console output is fine). Reasoning: Your network might train much slower on the tutor's system than on yours.
  - If you are aware that your network never stops training, please be honest and add a short statement saying so. Thank you!

## Problem 1 Mandatory Paper (1P)

Read this week's mandatory paper[1]. Which anomaly is the WS-353 dataset suffering from? Answer in one sentence.

## Problem 2 Softmax (4P)

The softmax function is a mapping from a vector $\mathbf{z} \in \mathbb{R}^n$ to a vector $\mathbf{y} \in \mathbb{R}^n$ and is defined as:

$$\mathbf{y} = (y_1, \ldots, y_n) = \text{softmax}(\mathbf{z}) \quad \text{where} \quad y_i = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)}, \forall i = 1, \ldots, n$$

### Problem 2.1 Comparison to other Activation Functions (1P)

In the domain of neural networks, the softmax function is oftentimes considered to be an activation function. What is the fundamental difference of the softmax activation function compared to conventional activation functions such as tanh or sigmoid? State your answer in one sentence.

### Problem 2.2 Mathematical Properties (3P)

Prove that the following properties of the softmax function hold:

a) $\forall y_i : 0 < y_i \leq 1$

b) $y_1 + y_2 + \ldots + y_n = 1$

c) $$\frac{\partial y_i}{\partial z_k} = \begin{cases} y_k(1 - y_k) & \text{for } i = k, \\ -y_i y_k & \text{for } i \neq k. \end{cases}$$

[1] https://www.aclweb.org/anthology/W16-2506

## Problem 3 Word Similarity with Word Embeddings (5P)

### Problem 3.1 Setup (0P)

For this home exercise, you will need:

- the gensim python library[2] (included in the docker file)
- the binary pretrained 300-dimensional word2vec embeddings from Google[3], extracted
- the WS-353 dataset[4], a classic word similarity dataset

### Problem 3.2 Word Similarity (5P)

In this task, you will evaluate how well the pretrained word2vec embeddings perform in the word similarity task on the WS-353 dataset. Gensim provides such a functionality with the `evaluate_word_pairs` method, but we will take the manual route in this task.

a) In the `combined.tab` file from the WS-353 dataset, each row contains two words and a mean similarity score assigned by humans. The three values are separated by tab characters (`\t`). Write a python method which reads the dataset into an appropriate format. (1P)

b) Load the pretrained binary word2vec embeddings with gensim, then compute the pairwise word similarity between each word pair using gensim's `similarity` method. (2P)

Hints:

- `https://radimrehurek.com/gensim/models/keyedvectors.html`
- If you run into memory errors because the embeddings are too large, you may use the `limit` parameter of the `load_word2vec_format` method.

c) Spearman's rank correlation coefficient[5] is a typical choice for measuring the ranking of two variables. Compute the coefficient between the values assigned by humans and your results from b) using scipy. Explain in two sentences what your resulting coefficient means. (2P)

[2] gensim: `https://radimrehurek.com/gensim`

[3] Pretrained word2vec embeddings: `https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing`

[4] WS-353 dataset: `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/`

[5] Spearman's rank correlation coefficient: `https://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient`