# Exercise 2 - To Embed or Not to Embed ...

*Building Word Embeddings with PyTorch/TensorFlow*

### Deadlines

Deadline for Exercise 2 is **26.10.2020, 00:15 CET**
Deadline for the peer review is **02.11.2020, 00:00 CET.** You will find instructions for the peer review process at the end of this document.
Deadline for feedback to your peer reviewers is **06.11.2020, 00:00 CET**

### Learning goals

This exercise introduces you to PyTorch (you're welcome to use TensorFlow as well) and how we can use it to create our own corpus-specific word embeddings. By completing this exercise you should …

- … understand the basic building blocks for training models in PyTorch/TensorFlow.
- … understand what word embeddings are and how you train them.
- … think of ways how to evaluate word embeddings.

Please keep in mind that you can always consult and use the exercise forum if you get stuck (note that we have a separate forum for the exercises).

### Deliverables

We encourage you to hand in your solutions as a Colab-Notebook. **Download your notebook as a .ipynb file**. That way your reviewers can view and execute your code. Or can view your already executed code.

Please hand in your code and your lab report. Hand in the following files and name them exactly in the following fashion:

- ex02_wordembeddings.ipynb
- ex02_labreport.pdf

zip it and name the zip-folder *ex02_ml4nlp1.zip*.

The .ipynb files contain your well documented AND EXECUTABLE code. We recommend you use Google's Colaboratory, where you have access to GPU time.

If you prefer to solve the exercise on your own computer please submit a zip-folder containing the required files:

- ex02_wordembeddings.ipynb
- ex02_labreport.pdf

We assume that the data files are in the same folder as the scripts, e.g.

- ex02_wordembeddings.ipynb
- scifi.txt
- tripadvisor_hotel_reviews.csv

Please submit the lab report in PDF format. The lab report should contain a detailed description of the approaches you have used to solve this exercise. Please also include results. In this exercise description, we highlight places in green where we expect a statement about an issue in your lab report.

Please note:

- Your peers need to be able to run your code. If it does not work, you will not be able to obtain the maximum number of points.
- DO NOT submit the data files!

### Data

You will work with the complete the "Trip Advisor hotel reviews" and "Sci-fi stories" for this exercise. Download the data from the Exercise 2 section in the OLAT. The folder contains the files "tripadvisor_hotel_reviews.csv" and "scifi.txt".
The files are also hosted under the following links:

- tripadvisor_hotel_reviews.csv
  https://drive.google.com/file/d/1ihP1HZ8YHVGGIEp1RHxXdt3PPIi12xvL/view?usp=sharing
- scifi.txt
  https://drive.google.com/file/d/10ehW4jZND3QA29v9aNboYUett5-swuNe/view?usp=sharing

The above common URL will help you link the data directly to your Colab notebook as you already did it in Exercise 1.

### Part 1 - Train your CBOW embeddings for both datasets

Go to this Colaboratory Python Notebook and complete the missing code in the exercise section (homepage from where we took it), the CBOW model. Change the code so it takes the Trip advisor hotel reviews text as input and produce word embeddings for the hotel reviews. **Important:** Use an OOP-oriented approach as close as possible to the one presented in Chapter 3 from Rao and McMahan. Concretely speaking, build the classes for the vectorizer, data loader, etc. according to their decomposition of the problem. It is up to you to decide on the preprocessing (removing punctuation, lower-casing, numbers etc.).

1. Describe your decisions in the lab report.

You will train **two** models: CBOW2 with a context width of 2 (in both directions) for **Hotel Reviews dataset and Sci-Fi story dataset.**

(**optional but recommended**) - Train CBOW5 with a context width of 5 (in both directions) and 50 dimensions. Are predictions made by the model sensitive towards the context size?

When training on small datasets, the number of iterations might have an influence on quality. Make sure you run enough epochs. Tune the embedding size for better results. This paper (here) is an interesting work in this domain.

### Part 2 - Test your embeddings

Word embeddings are not easy to evaluate automatically without suitable test sets. For this exercise, we inspect them manually to get a feel for whether they capture what we think they should capture. Computing the nearest neighbours (see below) for a word allows to get an intuition on the semantic vector space. Do the following for CBOW2 and, optionally, for CBOW5:

2. For nouns, verbs, and adjectives, take 3 examples each with different corpus frequencies for both the corpus (rare and frequent words) and look at the 5 closest words predicted. List them in your report and comment on the performance of your model: do the neighbours the model provides make sense? Discuss
3. What are the 2 common words in both the corpus? And what are the top 3 closest words predicted for them?
4. Do you notice the influence of iterations on quality? What is your best embedding size? Discuss
5. Do you think that certain words and their meanings have strong domain bias?
6. What are the differences between CBOW2 and CBOW5 (if trained)? Can you "describe" them?

### Function for nearest neighbour computation

```
import torch.nn as nn

def get_closest_word(word, topn=5):
    word_distance = []
    emb = net.embeddings_target
    pdist = nn.PairwiseDistance()
    i = word_to_index[word]
    lookup_tensor_i = torch.tensor([i], dtype=torch.long)
    v_i = emb(lookup_tensor_i)
    for j in range(len(vocabulary)):
        if j != i:
            lookup_tensor_j = torch.tensor([j], dtype=torch.long)
            v_j = emb(lookup_tensor_j)
            word_distance.append((index_to_word[j], float(pdist(v_i, v_j))))
    word_distance.sort(key=lambda x: x[1])
    return word_distance[:topn]
```

“net” above corresponds to the CBOW class in the Colab notebook. Note that you might have to adapt the code above, so it fits your initialization of the model.

### Important

Please make sure you run your code on Google Colab with GPU selected. Make the GPU selection from “Edit” → “Notebook Settings” and then make the GPU hardware accelerator.

## Peer Review Instructions

First: go to www.eduflow.com/join and join the class with the code **42D2XJ**. Important: Register with the E-mail address you use for OLAT. You might have already done this for the first exercise.
As soon as the deadline for handing in the exercise expires you will have time to review the submissions of your peers. You need to do **2 reviews** to get the maximum number of points for this exercise.
Here some more rules:

- If you do not submit 2 reviews, the maximum number of points you can achieve is 0.75 (from a total of 1).
- Please use full sentences when giving feedback.
- Be critical, helpful, and fair!
- **All reviews are anonymous: Do not put your name into the python scripts, the lab report or the file names.**
- You must also give your reviewers feedback. The same criteria as above apply.
- Students that consistently provide very helpful feedback can be awarded with a bonus in case they earned less than 6 points in total. Ways to obtain points are thus the following:
    - 5 exercises = 5 points
    - 1 presentation or research paper dissection = 1 points
    - consistently good reviews = 1 point

**Groups:**

- You can create groups of two to solve the exercise together.
- Both students should submit the solutions separately.
- When submitting the exercise, write a small post in the “Groups”-thread in the exercise forum on OLAT to notify the instructors about the group.
- As a group member, you still have to review two submissions with your own eduflow account. However, you may work together in the group to write all 4 reviews.