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

Building Word Embeddings with PyTorch

Deadlines

Deadline for Exercise 2 is **29.10.2023, 23:59 (Zurich Time)**.
Deadline for the peer review is **06.11.2023, 23:59 (Zurich Time).** You will find instructions for the submission and peer review process at the end of this document.

Learning goals

This exercise introduces you to PyTorch 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.
- … understand what word embeddings are and how you train them.
- … think about how one can 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 should contain your well documented AND EXECUTABLE code.

We recommend you use Google's Colaboratory, where you have access to GPU time. You can try to solve the exercise on your own computer. However, be aware that training without a GPU may take a long time.

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 a reduced version of "Trip Advisor hotel reviews" and "Sci-fi stories" for this exercise. Download the datasets from the material folder in the exercise section of OLAT. The folder contains the files "tripadvisor_hotel_reviews.csv" and "scifi_reduced.txt".
The files are also hosted under the following links:

- tripadvisor_hotel_reviews_reduced.csv
- scifi_reduced.txt

The above URL can be used to load the data directly into your Colab notebook - as you already did in Exercise 1.

Part 1 - Train your CBOW embeddings for both datasets

Go to this Colaboratory Python Notebook (page from where we took it), create a copy of it and complete the missing code in the exercise section - which means implementing a CBOW model in PyTorch. Then, change the code so that it takes the Trip advisor hotel reviews text as input and produces word embeddings for the hotel reviews.

Make sure that the code is understandable by either using comments for classes and methods or by explaining the code in text cells of the notebook.

It is up to you to decide on the specific preprocessing steps (splitting sentences, tokenizing words, dealing with punctuation, character casing, removing stopwords (perhaps!?), etc.).

1. Describe your decisions (preprocessing, class structure) in the lab report.

You will train three models:

- CBOW2 with a context width of 2 (in both directions) for the **Hotel Reviews dataset.**
- CBOW5 with a context width of 5 (in both directions) for the **Hotel Reviews dataset.**
- and CBOW2 with a context width of 2 (in both directions) for the **Sci-Fi story dataset.**

In order to obtain useful embeddings without training too long, we recommend an embedding size of 50 and 12-15 epochs on the hotel reviews dataset. On the Sci-fi dataset 3 epochs are enough.

Train CBOW5 or the **Hotel Reviews dataset** with a context width of 5 (in both directions). Are predictions made by the model sensitive towards the context size?

(**optional read**) - This paper (here) provides an insight on how to choose a minimum embedding size while still obtaining useful representations.

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 us to get an intuition of the semantic vector space. Do the following for CBOW2 and, optionally, for CBOW5:

2. For the hotel reviews dataset, choose 3 nouns, 3 verbs, and 3 adjectives. Make sure that some nouns/verbs/adjectives occur frequently in the corpus and that others are rare. For each of the 9 chosen words, retrieve the 5 closest words according to your trained CBOW2 model. List them in your report and comment on the performance of your model: do the neighbours the model provides make sense? Discuss.
3. Repeat what you did in 2. for the Sci-fi dataset.
4. How does the quality of the hotel review-based embeddings compare with the Sci-fi-based embeddings? Elaborate.
5. Choose 2 words and retrieve their 5 closest neighbours according to hotel review-based embeddings and the Sci-fi-based embeddings. Do they have different neighbours? If yes, can you reason why?
6. What are the differences between CBOW2 and CBOW5 ? 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.

NOTE: ONLY SWITCH from CPU to a GPU instance **AFTER** you have confirmed that your training procedure is working correctly. This may require rerunning the setup and preprocessing again (if you don't store intermediate processing files), but this is a smaller issue than running out of GPU hours.
Please ensure you run your final code on Google Colab with GPU selected. Make the GPU selection from "Edit" → "Notebook Settings" and then use the GPU hardware accelerator.

Training on a T4-GPU may take up to half an hour per model. If you are inactive in the Colab environment for a certain period of time, Colab may regard your user session as idle and disconnects - which disrupts the training. Make sure to prevent that from happening manually, or use a half-automatic trick like this. To prevent costly computations if your colab runtime crashes, save the output of your cost-intensive cells (which you can later without rerunning that code cell).

Submission & Peer Review Guidelines:

Peer Reviews will be carried out on OLAT.
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 **3 reviews** to get the maximum points for this exercise.
Here are some additional rules:

- **All file submissions are anonymous (for peer review purposes): Do not write your name into the Python scripts, the lab report, or the file names.**
- **ONLY ONE** member of each team submits on OLAT.
- Please submit a zip folder containing all the deliverables.

Groups & Peer Reviews:

- You can create groups of up to three people to solve the exercise together. Each member should contribute equally!
- If you did not already work together for the previous exercise (or already submitted a post with the same team members), write a small post in the "Assignment Team Submission Thread" in the exercise forum on OLAT to notify the instructors about the group.
- Only **one team member submits the solutions.**
- Only the submitting team member will have access to the peer review, however, you should distribute the workload evenly!
- If you do not submit 3 reviews, the maximum number of points you can achieve is 0.75 (out of 1).
- Please use full sentences when giving feedback.
- Be critical, helpful, and fair!
- Please answer all the review questions of the peer review.