Prof. Dr. Andreas Hotho, Albin Zehe, Jonas Kaiser
Data Science Chair

19./20.06.2024

7. Assignment in “Machine Learning for Natural Language Processing”

Summer Term 2024

1 General Questions

1. What are the differences between a recurrent neural network that processes inputs of length $l = 3$ and a multilayer perceptron (fully-connected network) that has three layers?

? Something to think about

2. Two problems in Machine Translation that still remain with biRNN Seq2Seq models are a) unknown words (e.g., names) and b) forgetting/duplicating parts of the input. Think about possible solutions to these problems!

? Something to think about

3. Propose a neural network architecture that would be suited for generating image descriptions, that is, given an image as input, creates a sentence describing the content of the image! Provide reasoning why your architecture is a good choice for this task.

2 Beam Search

In the lecture, you learned that Beam Search can sometimes lead to better translations than simple Greedy Search, where the decoder always chooses the locally most likely token.

To show that this can indeed happen, look at the following situation: Given an Encoder-Decoder network trained to translate from English to German. The encoder has read the sentence “I won’t tell you nothing!” and produced an internal representation h that encodes this sentence. This representation is now fed to the decoder and a translation is generated (see Figure 1). The decoder will return the following probabilities for output tokens:

$$
\begin{gathered}
P(\text{Ich}|\text{<s>}) = 0.9 \\
P(\text{Er}|\text{<s>}) = 0.01 \\
P(\text{verrate}|\text{<s>,Ich}) = 0.3 \\
P(\text{verrate}|\text{<s>,Er}) = 0.05 \\
P(\text{sage}|\text{<s>,Ich}) = 0.5 \\
P(\text{sage}|\text{<s>,Er}) = 0.08 \\
P(\text{euch}|\text{...,verrate}) = P(\text{euch}|\text{...,sage}) = 0.3 \\
P(\text{dir}|\text{...,verrate}) = P(\text{dir}|\text{...,sage}) = 0.45 \\
P(\text{nichts}|\text{...,euch}) = P(\text{nichts}|\text{...,dir}) = 0.35 \\
P(\text{nicht}|\text{...,euch}) = P(\text{nicht}|\text{...,dir}) = 0.4 \\
P(\text{!}|\text{...,nichts}) = 0.8 \\
P(\text{!}|\text{...,nicht}) = 0.1 \\
P(\text{nichts}|\text{...,nichts}) = 0.01 \\
P(\text{nichts}|\text{...,nicht}) = 0.2
\end{gathered}
$$

1. Perform a greedy search to get the output of the decoder network!
2. Perform a beam search with $B = 2$ to get the output of the decoder network!
3. Compare the two outputs.

3 Python

3.1 Implementing Neural Networks Part 5 — Embeddings

In this assignment, you will implement a neural network “library” yourself, using `python` and `numpy`.

This week, you will use your implementation to train your own word embeddings on a small corpus.

For this, download the dataset of English sentences from `https://data.deepai.org/text8.zip` and extract the .txt file.

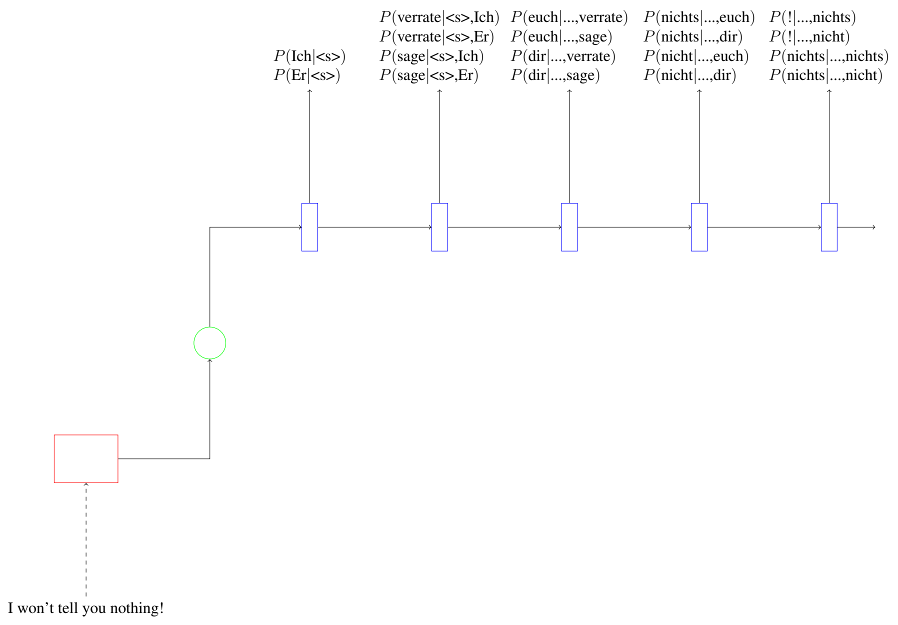

Figure 1: Visualisation of the Encoder-Decoder network used in this task

1. Extract training samples from the given corpus: Slide a context window over the corpus, creating tuples of the form $(\underbrace{w_{i+2}}_{\text{center word}}, \underbrace{(w_i, w_{i+1}, w_{i+3}, w_{i+4})}_{\text{context words}})$
2. Implement a simplified version of the skip-gram model: Your model should have the following layers:
   - Input: a 1-hot vector representing the center word of each sample
   - Hidden Layer: a layer of size 32
   - Output: a layer of size $|V|$, where V is the vocabulary of the corpus
3. Train the network on your samples: Use the center word as the input and train the network to predict the "4-hot" vector of the context words, that is, a vector that is zero everywhere except at the positions corresponding to the output words. (Note that this is not how the original Word2Vec is calculated. We change the procedure as you have not implemented the Cross Entropy loss function.)
4. After training, where will the embeddings be found?

Feel free to try different optimisers, activation functions, layer sizes, maybe apply Dropout, and look into the resulting word embeddings!