Default Project 3 - DD2424, 2025

Teachers and TAs of DD2424 2025 and earlier

1 NLP project

In this default project you should upgrade Assignment 4 from a simple vanilla RNN to a deeper LSTM. As a reference and some tips of implementation please check out Shakespeare Text Generation (using RNN LSTM)

1.1 Basic project to get E

For the basic assignment you should select a distinctive set of data to train on such as:

- The poems of Emily Dickinson: Project Gutenberg's Poems: Three Series, Complete, by Emily Dickinson
- The plays of Shakespeare, a subset of the plays is available here shakespeare.txt.
- Or perhaps don't focus on literature but on a programming language instead!

Project Gutenberg is a good source for books that can be downloaded.

Also in this upgrade you should construct training, validation and test sets so that performance can be measured more quantitatively. Besides the prediction loss you should also think of other metrics to measure the quality of generated text. To complete the basic assignment you should:

- Train an RNN baseline (use a pytorch/tensorflow implementation) on the dataset you use and compare at least to both a one and two layer LSTM both qualitatively and quantitatively.
- Generate diverse sequences from the training data to construct the training batches.
- Investigate how increasing the number of the nodes of the hidden state increases or decreases performance.
- Investigate the influence of different training parameters such as batch size and learning rate. You can for example perform a grid search or random search to find the optimal training parameters.
- During text generation you should scale the output probabilities with a temperature to generate more or less predictable passages of text. Also implement and investigation *Nucleus Sampling* as described in The Curious Case of Neural Text Degeneration by Ari Holtzman et al., ICLR 2020.

- When assessing the quality of the generated text of greater length use a variety of metrics. A couple of examples are percentage of correctly spelt words generated, the frequency of n-grams (with $n = 2, 3$ that occur in the generated text also occurring in the reference training text etc.

1.2 Extending the basic project to get a higher grade

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade.

1.2.1 From E $\rightarrow$ D/C

If you are aiming for either a D or C then here are a couple of extensions to the basic project you could apply and investigate:

- Compare the effect of GRU Vs LSTM layers
- More thoroughly investigate if more depth helps performance. If you begin to overfit during training add dropout layers and also add layer normalization to help with gradient flow.
- Can you mimic some of the visualizations in the section *Visualizing the predictions and the "neuron" firings in the RNN* in The Unreasonable Effectiveness of Recurrent Neural Networks.
- Instead of training RNN, LSTM and GRU on a single dataset, train them on two distinct datasets (for example Emily Dickinson and Shakespeare). First, check the performance on both datasets if you train on one of the datasets first and then on the other one. Second, create a mixed dataset which contains samples from both initial datasets and test the performance on the mixed and the initial datasets.

You are, of course, encouraged to come up with your own extensions and investigations of the network structure. But do remember to get explicitly write them down in your project proposal and get them vetted by a TA.

1.2.2 From E $\rightarrow$ B/A

If you are aiming for a B or A then here are some possible extensions from the basic E grade project you could investigate:

- **Upgrade your input tokens**

  Use words as the basic entry in the network and use a standard word embedding such as word2vec etc or Glove. Next investigate Byte-Pair Encoding (BPE) tokenization. Many LLMs use BPE tokenization, or some similar approach, as opposed to words or individual characters as the basic input to their networks. Hugging Face has a nice tutorial Byte-Pair Encoding tokenization explaining BPE. Summarize how the quality of the generated text increases/decreases when using word embedding or BPE tokenization compared to character level inputs. There is also a nice github repository by Andrej Karpathy minbpe which you can use to help guide your efforts.

- **Replace the RNN architecture with a Transformer**

  For this extension you would write a simple Transformer network similar to a scaled GPT to predict the next character in the sequence. You can use this excellent repository by Andrej Karpathy nanoGPT to help guide your work.

- **Explore the data-augmentation for NLP**

  Investigate using data-augmentation methods see here Data Augmentation in NLP: Best Practices From a Kaggle Master for a summary of approaches and a link to a library to help with the augmentations.

You are, of course, encouraged to come up with your own extensions. But do remember to explicitly summarize these extensions in the project proposal and get them vetted by the TA.