NLP using RNNs, LSTMs, and Transformers
Project Proposal

DD2424 Deep Learning in Data Science
KTH Royal Institute of Technology

David Alm, 020207-4795, davialm@kth.se
Theofanis Georgakopoulos, thegeo@kth.se
Antony Zhang, 011231-8159, antonyz@kth.se

April 23, 2025

1 Problem Description, Data and Software

We will be doing the 3rd default project on natural language processing (NLP) in which we build on the vanilla RNN developed in Assignment 4 to a deeper LSTM. The LSTM was orginally proposed in the 1997 paper Long Short-Term Memory by Hochreiter & Schmidhuber.

We will use The Complete Works of William Shakespeare for our data. Specifically, we shall use the UTF-8 encoded plain text version which can be downloaded from Project Gutenberg. We choose this dataset because of its size, roughly 5 million characters. After some basic preprocessing needed for the data, we will be splitting into a training set of 70%, a validation set of 15% and a test set of 15%

Taking inspiration from the example project ”Shakespeare Text Generation (using RNN LSTM)”, we shall begin by building a baseline RNN using TensorFlow. Next, we shall make our own implementation of both a one and a two layer LSTM and compare it to this baseline RNN both quantitatively (e.g. loss) and qualitatively (e.g. examining synthesized text).

We will also explore the best way to go about generating training batches from the training data as this is not a trivial problem in NLP.

Continuing, we shall use TensorFlow (for faster implementation) to examine the effects of the number of nodes of the hidden state vector, varying training hyper-parameters, different ways of adding regularization (e.g. dropout, early stoppage, L^2), and using both temperature and Nucleus Sampling (as described in the paper The Curious Case of Neural Text Degeneration by Ari Holtzmann et al., ICLR 2020.)

When we assess the quality of generated text of greater length, we shall use a variety of metrics. Namely, the percentage of correctly spelt words generated, the frequency of n-grams (with $n = 2, 3$) that occur in the generated text also occuring in the reference training text, and the number of unique n-grams divided by the total number of n-grams to evaluate the diversity.

Moving on, we shall consider upgrading our input tokens. Specifically, we shall use words as the basic entry in the network and use a standard word embedding such as word2vec or Glove. We shall also investigate Byte-Pair

Encoding (BPE) tokenization. We shall compare performance, both quantitatively and qualitatively, to the character level based models used earlier.

Moreover, we shall investigate using a Transformer architecture. We shall make our own implementation and compared its performance to our previous models. Replace the RNN architecture with a Transformer using nano-GPT.

Finally, we shall also explore how we can boost data efficiency by data-augmentation for NLP. We shall rely on the blog post Data Augmentation in NLP: Best Practices From a Kaggle Master for ideas on how to implement it.

2 Project Aims

Our group aims to achieve the grade A. We believe the following milestones are reasonable for achieving the respective letter grades.

- E grade: Sufficient solutions to the tasks up until upgrading the input tokens (exclusive), a well-written report and video presentation.
- D-C range: E grade and sufficient solution to either implementing a Transformer network, using words as the basic entry to the network and standard word embedding, or BPE tokenization.
- B-A range: D-C range and one more sufficient solution to either implementing a Transformer network, using words as the basic entry to the network and standard word embedding, or BPE tokenization.

The specific skills/knowledge that each group member aims to acquire from this are the following:

David Alm: Learn about LSTMs, how to implement them, what they excel at, and what is important for them to perform well. I am also excited to learn about word embeddings, tokenization, transformers and best-practices for data-augmentation.

Theofanis Georgakopoulos: To learn further about RNNs, LSTMs using backpropagation through time and expand my knowledge on Transformer architectures, word-embeddings and tokenization.

Antony Zhang: Aims to dive deeper into the theory behind RNN and LSTMs, implement them as well as getting more familiar with using TensorFlow.

We finally propose some further relevant references:

References

[1] Graves, A. (2013). Generating Sequences With Recurrent Neural Networks

[2] Hochreiter & Schmidhuber (1997). Long Short-Term Memory. Neural Computation. Holtzman et al. (2020). The Curious Case of Neural Text Degeneration. ICLR. Extensions

[3] Vaswani et al. (2017). Attention Is All You Need. NeurIPS.