SENTIMENT ANALYSIS SU DATASET IMDB

Matteo Galiazzo
Dipartimento di Informatica - Scienza e Ingegneria
Università di Bologna
matteo.galiazzo@studio.unibo.it

January 23, 2025

ABSTRACT

TODO

1 Introduction

This project presents an implementation of sentiment analysis on the IMDB movie reviews dataset using deep learning approaches. The project explores two different Long Short-Term Memory (LSTM) architectures: a standard LSTM and a bidirectional LSTM model. The implementation leverages TensorFlow and Keras for model development and training, with the goal of accurately classifying movie reviews as either positive or negative.

1.1 Recurrent Neural Networks

The problem of using a MLP or CNN to process text is the fact that we have a fixed input. This means that if we use one of this architectures we are constrained by a fixed window for our input. For solving this problem scholars introduced Recurrent Neural Networks (RNNs). In a RNN we have a sequence of words $x_1, ..., x_n$. To process this sequence we inject x_1 in the hidden layer and output o_1. The information of x_2 is processed together with the output of the previous computation. So in principle when you compute the output of x_n you take into consideration all the previous output. This way the sequence length is independent from the network structure.

Two very common memory cells for RNNs are Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU). RNNs have a problem. They can only see the past of the sequence.

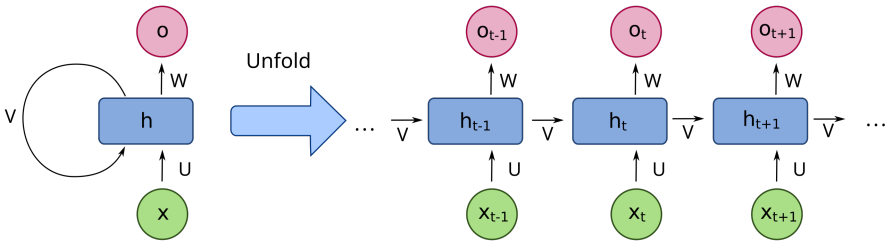

Figure 1: A compressed (left) and unfolded (right) RNN

Abstractly speaking, an RNN is a function f_θ of type $(x_t, h_t) \to (y_t, h_{t+1})$, where:

- x_t: input vector.
- h_t: hidden vector.

- y_t: output vector.
- θ: neural network parameters.

It's a neural netowrk that maps an input x_t into an output y_t, with the hidden vector h_t playing the role of "memory", a partial record of all previous input-output pairs. At each step, it transforms input to an output, and modifies its "memory" to help it to better perform future processing.

The figure 1 may be misleading to many because practical neural netowrk topologies are frequently organized in "layers" and the drawing gives that appearance. However, what appears to be layers are, in fact, different steps in time, "unfolded" to produce the appearance of layers.

We can combine two RNNs with one which gets an inverted input to see both the past and the future of the sequence.

We can combine two RNNs, one processing the input sequence in one direction, and the other one in the opposite direction This is called a bidirectional RNN, and it's structured as follows:

- The forward RNN processes in one direction: $f_\theta(x_0, h_0) = (y_0, h_1), f_\theta(x_1, h_1) = (y_1, h_2), ...$
- The backward RNN processes in the opposite direction:
  $f'_{\theta'}(x_N, h'_N) = (y'_N, h'_{N-1}), f'_{\theta'}(x_{N-1}, h'_{N-1}) = (y'_{N-1}, h'_{N-2}), ...$

The two output sequences are then concatenated to give the total output: $((y_0, y'_0), (y_1, y'_1), ..., (y_N, y'_N))$. Bidirectional RNN allows the model to process a token both in the context of what came before it and what came after it.

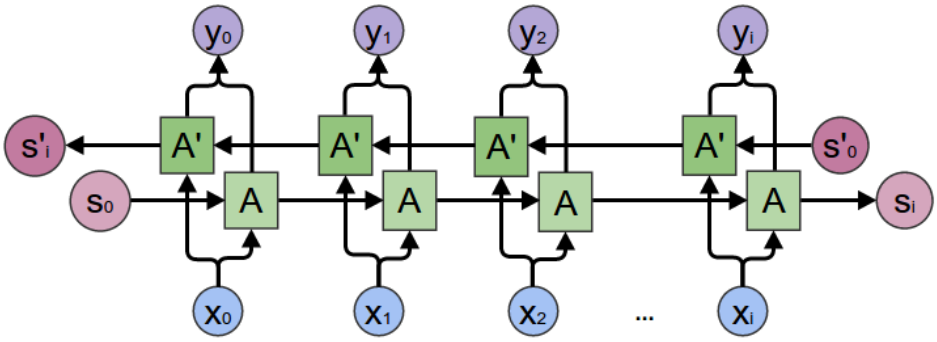

Figure 2: An unfolded bidirectional RNN

2 Technical implementation

2.1 Dataset

The IMDB dataset is a widely used benchmark for sentiment analysis tasks. It contains 50000 movie reviews from IMDB, allowing no more than 30 reviews per movie. The dataset contains an even number of positive and negative reviews, so randomly guessing yields 50% accuracy. Also, neutral reviews are not included in the dataset [1].

2.2 Data preprocessing

The implementation includes several preprocessing steps. The first one is data preprocessing. Data preprocessing is important because raw textual data often contains noise, suck as typographical errors, inconsistent formatting and irrelevant symbols. Preprocessing techniques aim to transform raw text data into a clean format that facilitates effective feature extraction and model training.

In this case I cleaned the dataset by removing:

- Links.
- Html tags.
- Double whitespaces and punctuation.

In this case I cleaned the dataset by removing: links, html tags, double spaces, punctuation and stopwords. The stopwords were removed by using NLTK (Natural Language ToolKit).

References

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.