Generating and Classifying Fake News

Aleksander Fuchs

January 2025

1 Introduction

With AI becoming more advanced in text generation, fake news is easier to create and harder to detect. AI-generated articles can look just as real as human-written ones, making misinformation a growing problem.

This project focuses on two things: generating fake news and classifying news as human- or AI-written. First, we use deep learning to generate realistic fake news. Then, we train a 1D CNN model to classify news articles based on whether they were written by a person or AI.

The source code is available at `https://github.com/Ojkee/fake-news-generator/tree/main`

2 Datasets

Two datasets [1][2] were used for training. Each contained over 40,000 news paragraphs, evenly split between real and fake news.

Random samples from both datasets, including both fake news and legitimate articles, were processed through QuillBot [4]. The tool classified all paragraphs as human-written.

The datasets included the following columns:

- Title
- Text
- Additional columns (filtered out)

Each paragraph was processed with a reduced vocabulary of 91 characters. More details are provided in later sections

3 Models

3.1 Generative Models

In this project, nine models with six different architectures were developed for the task of generating character-by-character text [3]. All models share the same

basic architecture, consisting of an LSTM layer followed by a dense layer with a softmax activation function.

The models differ in various configurations, but the core structure remains the same.

3.1.1 Basic Model

The first model is a standard LSTM with 256 units, followed by a dense layer with softmax activation. The output layer has a size of 91, corresponding to the size of the vocabulary. This model serves as the baseline for text generation.

3.1.2 Dropout Model

This model includes a dropout [5] layer between LSTM and Dense layer to prevent overfitting. This layer randomly sets a fraction of input units to 0 during training.

3.1.3 Batch-normalization Model

This model has additional Dense layer and Batch-normalization [6] layer in the middle, which helps stabilize the training process and accelerates convergence.

3.1.4 Bidirectional Model

This model is roughly the same as the above model, but the LSTM layer is set as Bidirectional [7].

3.1.5 Double LSTM Model

The Double LSTM model consists of two LSTM layers stacked on top of each other. The first LSTM layer has 256 units and is followed by another LSTM layer, with 128 units. The model is designed to capture sequential dependencies more effectively by allowing the second LSTM layer to refine the output of the first layer.

3.1.6 Model with Attention Mechanism

This model incorporates an attention [8] mechanism on top of the LSTM layer, allowing the model to focus on different parts of the input sequence when generating text.

3.1.7 Other Models

Other three models were the two batch normalization models and one attention model mentioned above trained on slightly different data.

3.2 Classifying Models

The classification model is based on a 1D Convolutional Neural Network (CNN).

It begins with an embedding layer that maps input tokens into an 8-dimensional space. This representation is then processed by a convolutional layer with 8 filters and a kernel size of 5, using the ReLU activation function. A global max-pooling layer follows, reducing the dimensionality while retaining the most significant features. Finally, a dense output layer with a sigmoid activation function produces a binary classification decision.

The idea behind CNN will be discussed in following section.

4 Training and experiments

The training process consisted of two main stages: text generation and classification. Training Text Generation Models

4.1 Generating Models

Eight different models were trained using six unique architectures, as described in Section Models. Two of these models used identical architectures to previously tested models but were trained on longer sentences from a different dataset. Each model was trained to generate text character by character, with a vocabulary size of 91.

A major issue observed across all models was cyclic repetition. In many cases, after a certain point in a generated paragraph—or sometimes from the very beginning—the text would fall into a repetitive loop of 2-3 words, continuing until the end of the paragraph. This problem significantly impacted the readability and coherence of the generated text.

Further, while bidirectional LSTMs and attention-based models achieved the lowest possible cost function values, they largely failed to produce meaningful text. The majority of generated outputs were either empty or filled with repeated ¡pad¿ tokens, rendering them unusable.

To address these issues, two models were selected for training on a curated dataset, which ensured that input sequences contained at least 140 characters while targeting a generation length of 150. This adjustment minimized excessive padding, leading to more structured outputs and reducing repetitive loops.

The training was performed using only a CPU, with training times varying between 1.5 hours for the simplest model and just over 2 hours for the most complex architectures.

Figures below show the cost function over 20 epochs for one of the models (left) and for 5 epochs for attention and bidirectional models (right). The trend is similar across almost all models, with a steady decrease in loss over time. Each model started with a cost function value of approximately 4 at the beginning of training.

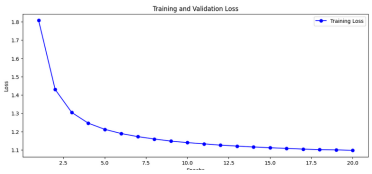

Figure 1: Fig 1. Training of almost every Generating Model

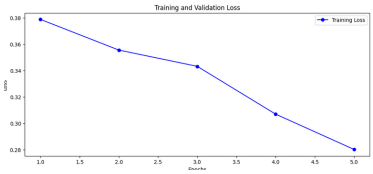

Figure 2: Fig 2. Training of Attention and Bidirectional Model

4.2 Constructing the Classification Dataset

Generating 6,000 sentences took approximately 9 hours, highlighting the computational cost of character-level text generation.

To train the classifier, the generated text samples were mixed with real data from the datasets [1][2]. This resulted in a balanced dataset containing both human-written and AI-generated text.

Interestingly, training the CNN classifier took only 2 seconds for 2 epochs, achieving 100

4.3 Evaluation and Observations

Training progress was monitored using loss and accuracy curves. Additionally, qualitative evaluation of generated sentences highlighted key weaknesses, such as repeated words and unnatural phrasing. These errors informed future model selection and hyperparameter tuning strategies.

5 Future Work

There are several directions to improve both the generation and classification aspects of this project. One key area is enhancing text generation by training and evaluating models on data produced by more sophisticated architectures, such as transformer-based models. In addition, efforts will be made to address cyclic repetition in generated text by exploring techniques such as penalty-based sampling or reinforcement learning strategies.

Another avenue for improvement is to experiment with word-by-word generation instead of the current character-level approach, which may lead to more coherent and structured outputs. In addition, fine-tuning classification models in adversarially generated fake news could improve robustness against more advanced AI-generated misinformation.

Beyond this, exploring multi-modal approaches—incorporating metadata such as publication sources, timestamps, or writing styles—could enhance classification accuracy. Finally, real-time fake news detection using streaming data and on-device inference for efficiency and scalability will also be considered

6 Ethic Statement

This project focuses on detecting AI-generated fake news, not producing them. While text generation was used for training, the goal is to improve methods for distinguishing human-written and AI-generated content. Misinformation is a serious risk, and this work aims to enhance detection techniques while adhering to responsible AI practices.

References

[1] A. Singh, *Fake News Classification*, https://www.kaggle.com/datasets/aadyasingh55/fake-news-classification

[2] E. Bozkus, *Fake News Detection Datasets*, https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets

[3] A. Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks*. http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[4] QuillBot AI Content Detector. https://quillbot.com/ai-content-detector

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. *Journal of Machine Learning Research*, 2014. https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

[6] S. Ioffe and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, in Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pp. 448-456, 2015. https://arxiv.org/pdf/1502.03167

[7] M. Schuster and P. Paliwal, *Bidirectional Recurrent Neural Networks*, IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, 1997. https://ieeexplore.ieee.org/document/650093

[8] A. Vaswani, N. Shazeer, N. Parmar, L. Uszkoreit, J. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is All You Need*, in Advances in Neural Information Processing Systems (NeurIPS), 2017. https://arxiv.org/abs/1706.03762